# Berliner Informatik-Tage

# bit '82

## Humboldt-Universität zu Berlin

Tagungsbericht

# BERLINER INFORMATIK-TAGE 1982

Wissenschaftliche Fachtagung zu Problemen der theoretischen und angewandten Informatik

28. Juni bis 2. Juli 1982

Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum

Herausgeber: Der Rektor der Humboldt-Universität zu Berlin
Redaktion: Dr.-Ing. W. Zschau, Organisations- und Rechenzentrum der Humboldt-Universität
DDR 1086 Berlin, PSF 1297
Druckgenehmigungsnummer: 8/950/82
Gesamtherstellung: Nationales Druckhaus Berlin
VOB National

Die wissenschaftliche Tagung "Berliner Informatik-Tage 1982", die vom 28. Juni bis zum 2. Juli 1982 an der Humboldt-Universität zu Berlin stattfand, war folgenden thematischen Schwerpunkten gewidmet:

Organisation und Technologie der Softwareproduktion,
Anwendung und Nachnutzung von Softwareprodukten,
Datenverwaltungssysteme sowie
Software zur Automatisierung der Rechenbetriebs.

Es wurden in Plenarvorträgen und in vier Sektionen Ergebnisse von Forschungs- und Entwicklungsarbeiten zu diesen Themenkomplexen von Referenten aus verschiedenen Bereichen der Volkswirtschaft, aus Akademieeinrichtungen und Hochschulen vorgestellt und in den Sektionen kritisch diskutiert.

Im vorliegenden Tagungsbericht werden die Manuskripte der meisten Beiträge veröffentlicht. Einige Referenten baten jedoch darum, von einer Veröffentlichung an dieser Stelle abzusehen, da Beiträge aus ihrer Feder zum gleichen Thema inzwischen in Fachzeitschriften erschienen seien. Wir haben diesem Wunsch entsprochen. Bedauerlicherweise konnten einige andere Referenten ihr Manuskript nicht rechtzeitig bereitstellen - auf ihren Beitrag mußte deshalb in diesem Bericht verzichtet werden. Alle eingereichten Manuskripte werden ungekürzt wiedergegeben.

Für ihren Anteil an der Gestaltung des vorliegenden Materials danken wir allen Autoren - für ihr aktives Mitwirken an der "bit'82" geht unser Dank an alle Referenten und Diskussionsredner.

Berlin, November 1982

Doz. Dr.sc.nat. Grabowski

Inhalt

Plenarvorträge

Köhli, S.
Zum Zusammenhang von Organisation und DV-Projektierung 7

Winks, H.
SPOOL - ein System zur Automatisierung von Produktionsprozessen an ESER-Rechnern; Entwicklungsstand und Entwicklungsrichtungen 40

Wenzlaff, B.
Gesellschaftliche und soziale Auswirkungen der EDV-Anwendung 58

Pilgrim, J.
Der EDV-Benutzer - Versuch einer Beschreibung und Definition 71

Herrlich, O.; Datering, R.
Anforderungen an Spezifikationsmittel im Programmentwicklungsprozeß 97

Bergholz, G.
Leistungsbewertung im Rechenbetrieb 105

Bazewicz, M.; Mika, T.
User conditions to software functions in distributed processing system /DPS/ 124

Heymer, V.
Tendenzen der Datenkommunikation 136

Kurzvorträge in den Sektionen

Sektion "Organisation und Technologie der Softwareproduktion

Schreiter, G.
Modulare Struktur eines Steuerprogramms für einen freiprogrammierbaren Steuerrechner 155

Sacklowski, U.
Gedanken zu Gestaltungsaspekten der Bildschirm-Interaktions-schnittstelle und ihren programmiertechnologischen Realisierungen 169

Detering, R.; Stantschev, V.
Vergleich rechnerunterstützter Spezifikationsverfahren 179

Pfeiffer, P.
Modulkopplung durch interne Datenkommunikation 187

Richter, E.; Tauscher, W.
Zur Verwendung von Datenabstraktionen unter besonderer Beachtung der Assembler-Programmierung 199

Trautloft, R.
Über einen Zugang zur automatisierten Auswahl physischer Realisierungen von relational definierten Bestandsdaten 209

Polze, C.
Experimente mit der Programmiersprache CDL2 215

Bothe, K.
Modulare Programmierung in Pascal (Kurzfassung) 220

Hohberg, B.
Syntaxorientierter Programmeditor für PASCAL 221

Kosciolowicz, R.
Konzeption eines Programmierlabors zur Unterstützung des modularen Programmentwurfs mit PASCAL 231

Horn, C.
Breitbandsprachen und Programmentwicklung 245

Nikulski, E.
TESYS - Ein technologisches System zur Software-Entwicklung 276

Bertholdt, D.
Ein allgemeines Steuerprogramm zur sequentiellen Auswertung von Dateien 281

Vack, G.
Verwendung des SM4-Makroassemblers als Übersetzer für einfache problemorientierte Sprachen 268

Sektion "Anwendung und Nachnutzung von Softwareprodukten"

Bauer, G.
Programmierung und Nachnutzbarkeit 284

Kieschke, R.
Auswahlkriterien für die Anwendung vorgefertigter Software der Dialogverarbeitung 291

Meuche, H.-F.; Wegener, E.
Anwendung des Mensch-Maschine-Dialoges für betriebswirtschaftliche Aufgaben 298

Kollár, L.
The Large Data Sets Processing Mathematical Software 308

Mikulecký, P.; Kelemen, J.
Supervising Expert Systems: An Attempt to Organize Certain Mathematical Software Intelligently 313

Schirrow, R.
Konzeption eines Auswertungssystems für das Krankenhaus Friedrichshain von SKR-Rechentechnik 326

Haberland, H.; Roscher, G.
Erfahrungen bei der Gestaltung und mehrfachen Nutzung vorgefertigter Grundsoftware in der technischen Vorbereitung 330

Aden, H.
Listgenerator LSDAT 342

Vierke, B.; Haagen, E.
Entscheidungstabellenvorübersetzer "WET" 352

Böhme, P.
Die Anwendung des Programmpaketes SYSDFV zur Simulation spezieller, durch Differentialgleichungen beschriebener Systeme 356

Fenske, D.; Meißler, H.-G.
Ein Konzept zum Einlesen eines Stromes von Lochbanddateien 362

Hagen, M.; Klaembt, E.; Schirmbacher, P.
Methodische Auswahl eines Datenauswertungssystems für den Einsatz in einer Informationszentrale 370

Wittig, W.
Elemente eines Planungsdialogs im Fernmeldewesen 379

Barrera, J.
Die Anwendung der Datenbanktechnik und des Mensch-Maschine-Dialogs in der zentralen staatlichen Planung von Kuba 384

Sektion "Datenverwaltungssysteme"

Pester, M.
Zur Parallelisierung von Datenverwaltung und -verarbeitung in Programmsystemen 408

Fellien, A.
Zur Konzeption des Informationssystems der Deutschen Staatsbibliothek 415

Wagner, K.
Zum Problem der Anwendung der Datenbanktechnologie 420

Bazewicz, M.; Debowy, J.
Software organisation and utilization method in a medical information system 428

Hagen, M.; Klaembt, E.; Schirmbacher, P.
Der Einsatz von automatisierten Informationsverarbeitungssystemen (AIVS) für den Leitungsprozeß 436

Fichtner, N.
Interdisziplinäre Zusammenarbeit bei der Anwendung der automatisierten Informationsverarbeitung im Gesundheitswesen und in der Medizin 446

Melzer, J.
Die Beziehung zwischen Struktur und Auswertung bei patientenbezogenen Datenbeständen 454

Jantke, R.
Das Datenverwaltungssystem MAS/D-OS 462

Sektion "Software zur Automatisierung des Rechenbetriebs"

Schreiber, E.
Programme zur Leistungsbewertung im DVZ Dresden 468

Sadowski, H.
Verfahren zur Automatisierung der Verarbeitungstechnologie 478

Kroß, G.
SPOOL 2.0 - System zur Unterstützung eines zeitweise bedienungsunabhängigen Betriebes an ESER-Rechnern 485

Meißler, H.-G.
SPOOLRJE - eine Fernverarbeitungskomponente des SPOOL-systems 491

Piens, C.
Effektivitätsuntersuchungen für das System SPOOL 3 498

Weickmann, C.
Gemeinsame Nutzung von TSO und SPOOL auf einer EDVA 503

Wörfel, F.
Aufbau und Handhabung des Systems der Katalogdateien für SPOOL 508

Anke, H.
Erfahrungen mit einem System zur Jobverwaltung 516

Bergholz, G.; Mrosko, G.
Simulationsmodell der Stapelverarbeitung im Multiprogramm-betrieb 524

Autorenverzeichnis 533

Prof.Dr.sc. S. Köhli
Ingenieurhochschule Köthen

# Zum Zusammenhang von Organisation und DV-Projektierung

## 1. Aufgaben, Rolle und wissenschaftliche Grundlagen der Organisation

### 1.1. Aufgaben und Rolle der Organisation bei der sozialistischen Rationalisierung

Unter der Tätigkeit der Organisation bzw. des Organisierens verstehe ich in Übereinstimmung mit den meisten Autoren die bewußte Herstellung eines Zustandes der Ordnung /1/, die Aufgaben der Strukturierung und Regelung eines sozialökonomischen Systems zur Erreichung der Betriebsziele /2//5//21/.

Moderne Auffassungen zur Organisation beziehen stets den Aspekt des zielorientierten Entscheidens im Sinne "Konzipierender Entscheidungen" ein, wobei " ... eine Handlungskette vorbereitet wird, deren Handlungsakte zur Erreichung der Zielstellung führen ..." /40/ /.
Die von der Erfüllung vorgegebener Ziele ausgehenden organisatorischen Entscheidungen haben primär die zweckmäßige Struktur und den Ablauf von Handlungsfolgen zum Gegenstand, woraus sich die zwei bekannten, konventionellen Hauptaspekte des Gegenstandes der Organisation ergeben, nämlich die Ablauf- und Strukturorganisation /19/ /21/.

Von dem Autorenkollektiv "Informatik und Automatisierung" wird der technologische Aspekt der Entscheidung" ... auf den Inhalt der einzelnen Handlungsakte", der organisatorische dagegen auf "das geeignete Zusammenwirken dieser Handlungsakte" (S. 44) bezogen /21/. Daraus kann die Bedeutung der Technologie und der Organisation in ihrem Zusammenwirken abgeleitet werden. Ich persönlich folgere daraus nicht nur eine gewisse strategische Rolle der Organisation zur Vorbereitung und Durchsetzung einer rationellen Technologie, sondern verbinde mit dem Wort Organisation stärker den Aspekt der Koordination von einzelnen und kollektiv tätigen Menschen. Noch stärker als bei der Technologie ist bei der

Organisation die Information Gegenstand, Mittel und Resultat der Organisationsarbeit, der stoffliche und energetische Aspekt eines Prozesses tritt bei ihr gegenüber der informationellen Verknüpfung menschlicher und maschineller Handlungen bzw. Operationen zurück. Damit wird bereits ein Unterschied der Organisation menschlicher Tätigkeiten gegenüber der Technologie deutlich:
die Technologie kann sich primär mit formalisierbaren Operationen beschäftigen /41/, die Organisation muß primär den Menschen mit seinen individuellen Fähigkeiten einbeziehen. Sie ist damit von der Qualifikation und Motivation der Menschen und spezifischen menschlichen Eigenschaften und damit stark von gesellschaftswissenschaftlichen Erkenntnissen abhängig, hat enge Beziehungen zur Soziologie, Arbeitspsychologie und -physiologie und stellt an diese neue Forderungen.
Organisation als wissenschaftliche Disziplin läßt sich damit primär den Wirtschaftswissenschaften zuordnen, trotz eines tendentiell stärkeren Einflusses der Informations- bzw. Organisationstechnik. Trotz oder gerade wegen der gegenüber der Technologie starken Abhängigkeit vom komplizierten Subjekt und wichtigstem Objekt der Organisation in Form der Arbeitskraft, stimme ich dem Autorenkollektiv unter Leitung von Fuchs-Kittowski zu, wenn es die Notwendigkeit betont, daß die Rationalisierung insgesamt auf wissenschaftlich fundierten Methoden aufbauen muß. Daraus folgt die Notwendigkeit, auch die Organisation wissenschatlich zu betreiben und ihr Verhältnis zu anderen Disziplinen zu untersuchen.

## 1.2. Wissenschaftliche Grundlagen der Organisationsarbeit und deren Bedeutung für die Informationsverarbeitung

Folgende Übersicht zeigt einige wichtige Ansätze organisationswissenschaftlicher Theorien /41/, die auch für die EDV-Organisation von Bedeutung sind:

(nach Tschirschwitz)

1. Klassisches Maschinenmodell der Organisation bzw. Taylorismus
2. Human Relations Approach (Organisations-Entwicklung)
3. Modell der kybernetischen Maschine bzw. Automat als Modell der Organisation
4. Modell des soziotechnischen Systems bzw. allgemeiner systemtheoretischer Ansatz
5. Organisation und kybernetisches System als Ausdruck der Subjekt - Objekt - Dialektik (Information und Wert, Aktion und Funktion, Kommunikation und Information, Rationalität und Humanismus im dialektischen Wechselverhältnis von Unterschied und Gemeinsamkeit),

Die skizzierte Entwicklung geht aus vom Taylorismus und mündet in modernen Informatik-Theorien.

Das klassische Maschinenmodell der Organisation bzw. der Taylorismus reduzierte den Menschen primär hinsichtlich seiner Fähigkeiten zur Durchführung von Routinetätigkeiten /24/ /41/ im Produktionsprozeß und leistete primär einen Beitrag zu Zeit- und Methodenstudien.

Viele EDV-Organisatoren werden bestätigen, daß das Modell der kybernetischen Maschine, der abstrakte Automat, oder Regelkreisvorstellungen für uns auch heute noch wichtige Denkmodelle sind /41/.

(Vgl. Vortrag von Fuchs-Kittowski)

Das soziotechnische System weist auch den EDV-Organisator darauf hin, daß auch bei aus seiner Sicht zweckmäßiger Lösung der technologischen, strukturellen und Steuerungsfragen große Auswirkungen aus dem psycho-sozialen Bereich kommen können, die Automatisierungsvorhaben stark beeinflussen /41/ /24/ /28/
Aus der Literatur über Mensch-Maschine-Kommunikationssysteme wird deutlich, daß die IV-Spezialisten schon über die zumutbare bzw. vertretbare Wartezeit am Bildschirm höchst divergierende Auffassungen vertreten. Alle genannten Ansätze haben bestimmte Aspekte hervor und haben auch für die EDV-Organisation m. E. noch methodologische Bedeutung.

Die an der Sektion Wissenschaftstheorie und Wissenschaftsorganisation der Humboldt-Universität vertretenen Auffassungen, wie sie im 5. Ansatz angedeutet sind, führen m. E. zu einem besseren Verständnis von Organisationsgrundtypen als Gegenstand und Resultat der Automatisierung /41/ /2/, wobei physionomische (nicht organisierte), Funktions- (organisierte), von Aktionssystemen (organisierenden) unterschieden werden.

"Organisationstechnologien enthalten die "... organisatorischen Anforderungen an die Gestaltung von Aktionssystemen auf der Orgware-Ebene ..." /41/, deren Inhalt Sinngebung und Bewertung u.a. von Funktionssystemleistungen in Aktionssystemen ist.

In diesem Sinne wird vom genannten Autorenkollektiv /2/ die Informatik als theoretische Grundlagenwissenschaft für die Organisation und die Informationsverarbeitung verstanden. Hierzu ergeben sich trotz meiner Zustimmung zu prinzipiellen Aussagen aus meiner Sicht eine Reihe von Problemen, Fragen und anderen Auffassungen, von denen ich hier nur einige nennen möchte:

1. Der sicher vom Anliegen her begründet weitgehend philosophisch-soziologisch orientierte Begriffsapparat und das hohe Abstraktionsniveau der Aussagen führen nach meiner Auffassung zu einer gewissen Verständigungsbarriere besonders zu Vertretern einzelner technisch-technologischer Teildisziplinen der Informationsverarbeitung und gegenüber vielen Praktikern.

2. Die bisherigen meist deduktiven, abstrakten Untersuchungen sollten meiner Ansicht nach durch anwendungsbezogene methodisch bzw. technologisch orientierte Untersuchungen, z.B. zur Reduktion von Aktionssystemen, zur Organisationstechnologie und entsprechende Erprobungen ergänzt werden.

3. Wenn diese Aussagen als Anleitung zum Handeln fungieren sollen, ist eine Vermittlung dieser Untersuchungsergebnisse gegenüber möglichst allen wissenschaftlich tätigen IV- und Orgspezialisten in Form entsprechender Seminare zweckmäßig, so daß es über die anwendungsorientierten Teildisziplinen zur Praxisanwendung kommt.
   Die angewandte IV wird sich auch künftig, trotz des direkten Bezuges und der Einordnung in Aktionssysteme primär nur mit der Schnittstelle zur semantischen Informationsverarbeitung und trotz ihres Applikationscharakters mit syntaktischer Informationsverarbeitung für definierte Nutzeraufgaben beschäftigen.

   Um Vorschläge zur Zusammenarbeit von Organisatoren und IV-Spezialisten abzuleiten, ist es notwendig, einige Aussagen zur Spezialisierung von Organisatoren zu formulieren, worauf ich im folgenden eingehen will.

Tabelle 3

Vorschlag für die berufliche Spezialisierung von Organisatoren und deren Aus- und Weiterbildung

| Organisat.-Spezial-Disziplin | Hochschulausbildg. einschl. Spezialisierg.u.Vertiefung | Postgraduale Spezialisierung |
|---|---|---|
| Wirtschafts-organisation | Wirtschaftsorganisator (Vertiefungsrichtung in d.Fachricht. Volkswirtschaft) | - f. Kombinate<br>- f. Staatsapparat |
| Leitungs-organisation | Leitungsorganisator (Vertief.-richtung bzw. primäres Ziel d.Fachrichtung Leitungswissenschaft) | - f.Wirtschaft u. Handel<br>- f. Staatsapparat, Gesundheitswesen u. sonst. |
| Produktionsorganisation<br><br>Bereichs-organisation | Studium als<br>1) Organisator<br>-Leitungs-,Verwaltungsorganisator,<br>Arbeitswissenschaftler<br>Wissenschaftsorganis.<br>2) Fachspezialist<br>z.B. für<br>-Forschungsökonomie<br>-Anlagentechnik<br>-Volkswirt | postgraduales Studium<br>- primär fachbereichsbezogen<br>- primär organisationsbezogen |
| Verwaltungs-organisation | Hochschulgrundstudium: Wirtschaftswissenschaften<br>Spezialisierung: Verwaltungsorganisation | - Vertiefung Org.-technik<br>- Methoden d. Verwaltungsorgan.<br>- WAO |
| WAO | Arbeitswissenschaften | - Leitungsorganis.<br>- Verwaltungsorga.<br>- Arbeitsorganis. |

Tabelle 3 Fortsetzg.

| Organisation Spezial-Disziplin | Hochschulausbildg. einschl. Spezialisierg.u.Vertiefung | Postgraduale Spezialisierung |
|---|---|---|
| Informations- bzw. EDV-Organisation | Wirtschaftswissenschaften/ Informationsverarbeitung (ASU-Kadergruppe IVa mit ca. 25% Inform.-verarb.-Anteil) "Angewandte IV" | - Leitungs-u. Verwaltungs-organisation Anwendungsgebiet |
| | Wirtschaftswissenschaften/ Leitungsorganisation | .Inform.-verarb. .Anwendungsgebiet .Verwaltungsorg. |
| | Wirtschaftswissenschaften/ Verwaltungsorganisation | .Inform.-verarb. .Leitungsorgan. .Anwend.-gebiet |
| | Informationsverarbeitung (Spezialisierung) IVS-Entwurf **DV-Projektierung f. TEVO** | .Leit.-organis. .Verw.-organis. .WAO .TEVO |
| | Angewandte IV auf techn. Fachgebiet (Kadergruppe IVa) | |
| Wissenschafts-organisator | Organisator (Wissenschaft) Humboldt-Uni | postgradual - Spezialisierung auf 1 Fachgebiet |
| | Forschungsökonom Leitungswissenschaftler (Leitungsorganisator) | - Spezialisierung auf Wiss.-Org./ Inform.-Verarb. |
| | Wissenschaftler auf dem Fachgebiet | - Spezialisierung auf Wiss.-Org./ Inform.-Verarb. |

## 1.3. Probleme der Spezialisierung von Organisatoren

Die Spezialisierung von Organisatoren ist eine Voraussetzung für die detaillierte Beschäftigung mit Fragen der jeweiligen Anwendungsdisziplin, um eine ausreichende Detailliertheit in der Auseinandersetzung mit dem Arbeitsgegenstand zu erreichen und um detaillierte praktische Erfahrungen im Berufsleben kumulieren und schöpferisch anwenden zu können. Zu dieser Frage gibt es weder in der Literatur, noch in der Praxis einen einheitlichen Standpunkt. Hierzu will ich meine persönliche Meinung aus der Sicht organisatorischer Überlegungen zur DV-Projektierung darlegen. Dabei stimme ich u. a. mit dem Autorenkollektiv des Leitfadens zur Organisationsprojektierung /3/ nur teilweise bei der Formulierung unterschiedlicher Gegenstandsbereiche der Organisation als Grundlage der disziplinären Spezialisierung und Arbeitsteilung überein, wie Tabelle 3 zeigt.

Die Organisationsprojektierung sollte meiner Auffassung nach nicht als eine berufliche Spezialisierungsrichtung aufgefaßt werden, sondern als eine vorhabenbezogene komplexe Organisationsaufgabe, bei der eine umfassende organisatorische Rationalisierung für ein definiertes Investobjekt, eine definierte Struktureinheit, einen definierten Prozeß interdisziplinär von unterschiedlichen Organisationsspezialisten und anderen Fachleuten gelöst wird.

Die klare Abgrenzung eines Organisatoren-Profils, das neben der dringend erforderlichen und von außen durchschaubaren Spezialisierung, gleichzeitig eine bestimmte Bildungsökonomie und die postgraduale Weiterbildung einbezieht, ist m. E. eine primäre Voraussetzung für die rationelle Organisation interdisziplinärer Arbeit, wie sie u. a. die Gestaltung automatisierter IVS darstellt.

## 2. DV-Projektierung als Mittel und Gegenstand der Organisation im Prozeß der Rationalisierung

### 2.1. DV-Projektierung als interdisziplinäre und organisatorische Tätigkeit (Mittel der Organisation)

Gehen wir von der in der Rahmenordnung der DV-Projektierung enthaltenen Begriffsdefinition aus, dann versteht man unter DV-Projektierung "die Arbeiten zur Bereitstellung der DV-Projekte ..." und unter diesen "... alle Projektunterlagen, die für die problembezogene Nutzung der DV-Technik erforderlich sind." Bekanntlich sind innerhalb der Einsatzvorbereitung, außerhalb der DV-Projektierung, einige Aufgaben, wie die Schaffung der personellen, technischen und organisatorischen Voraussetzungen, die primär als organisatorische Tätigkeiten angesehen werden müssen /37/.
Geht man von der Gliederung des Projektierungsprozesses nach den je DV-Projekt und Stufe erforderlichen Projektierungsaussagen aus, wie sie in der mit SCHNEIDER gemeinsam verfaßten Broschüre /37/ enthalten sind, so ergibt sich, daß zahlreiche Projektaussagen überwiegend bzw. teilweise organisatorischen Charakter aufweisen. Nur einige Projektaussagen, die primär die Implementierung betreffen, haben keinen bzw. kaum organisatorischen Charakter.

Insgesamt kann man davon ausgehen, daß einige Projektierungsstufen, nämlich E1, E2, E5, E6 als überwiegend organisatorische Tätigkeiten aufzufassen sind.
Damit gilt die These, daß DV-Projektierung stets Organisationsarbeit darstellt.

## 2.2. DV-Projektierung als Objekt der Organisation

DV-Projektierung stellt nicht nur selbst Organisationsarbeit dar, sondern ist auch gleichzeitig Objekt der Organisation /33/: sie wird organisiert. Typische diesbezügliche betriebliche Organisationsaufgaben sind u. a.:

1. Abgrenzung der Aufgaben der DV-Projektanten von denen der Fachbereiche
2. Ablauforganisation der DV-Projektierung
3. Unterstellung, Aufbau- und Themengruppenorganisation
4. Regelung der Testarbeiten, Bibliotheksführung, der Dokumentation, der Anwendung von Projektierungsmethoden, der Verteidigung, der Nachnutzung von Software usw..

Die Regelungen der Prozeßorganisation für die DV-Projektierung sind sehr fachspezifisch und haben aufgrund weitgehend einheitlicher Arbeitsmittel, nämlich der EDVA, in vielen Betrieben große Gemeinsamkeiten. Dies bewirkt, daß auch spezielle Orgware, das Know how zur Organisation der DV-Projektierung nicht nur in Form unterstützender Software nachnutzbar wird /33/. Damit haben auch wissenschaftliche Untersuchen zur Organisation der DV-Projektiering einen hohen Stellenwert. Bekannt ist allgemein, daß die Mehrfachnutzung von Produkten geistiger Arbeit, also auch von Projektierungsergebnissen, zum Hauptkettenglied der Rationalisierung wird. /33/ /30/
Daraus leitet sich eine besondere Rolle der Organisation im Rahmen einer Strategie zur Rationalisierung der DV-Projektierung ab, die die über- und zwischenbetriebliche Organisation einschließt. /30/ /33/

## 2.3. Zur Rolle der Organisation im Rahmen einer Strategie der DV-Projektierung

Nach seit 1978 vorliegenden Untersuchungen kann man von folgenden Hauptwegen der Intensivierung der DV-Projektierung ausgehen /33/:

1. Methoden, Technologien, Projektierungshilfsmittel (PH) einschließlich universeller Anwendungs-, Hilfs- und Standardprogramme

2. mehrfach nutzbare Gestaltung von DV-Lösungen

2.1. Einheitsprojekte

2.2. Teilweise variable Einheitsprojekte für einen definierten, bekannten Nutzerkreis

2.3. variable DV-Lösungen

3. Gesellschaftliche Organisation

3.1. Zentralprojektierung

3.2. Zentr. org. arbeitsteil. (koop.) DV-Projektierung

3.3. koop. org. arbeitsteil. DV-Projektierung

3.4. Konzentration der Kräfte und Mittel in den Formen: Zentralisation / Kooperation

4. Verbesserung der Arbeits- und Leitungsorganisation für Prozesse der DV-Projektierung

Daraus wird ersichtlich, daß die Hauptwege
- gesellschaftliche Organisation der DV-Projektierung mit Konzentration, Kooperation und Spezialisierung,
- betriebliche Leitungs- und Arbeitsorganisation

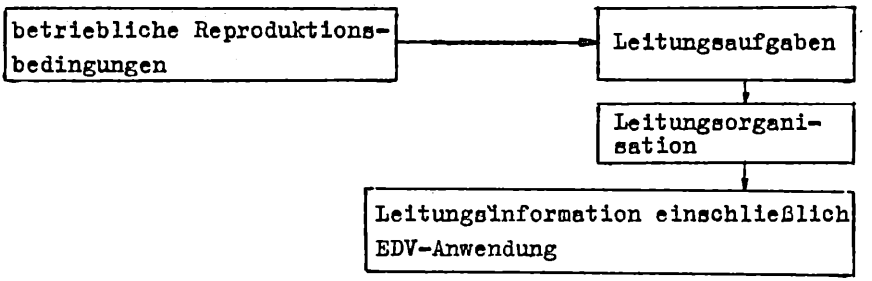
als Hauptwege einer Rationalisierungsstrategie angesehen werden müssen.
Zur wissenschaftlichen Untersetzung einer solchen Rationalisierungsstrategie gilt es jedoch, auch neue wissenschaftliche Fragestellungen zur Technologie der DV-Projektierung unter Vereinheitlichungsaspekten zu lösen, wobei ich hier nur einige Gedanken zu Vereinheitlichungsfragen und zur gesellschaftlichen Organisation darlegen kann.

## 2.4. Zum Zusammenhang von betrieblichen Reproduktionsbedingungen, Leitungsorganisation, Leitungsinformationssystem und Rationalisierung der DV-Projektierung

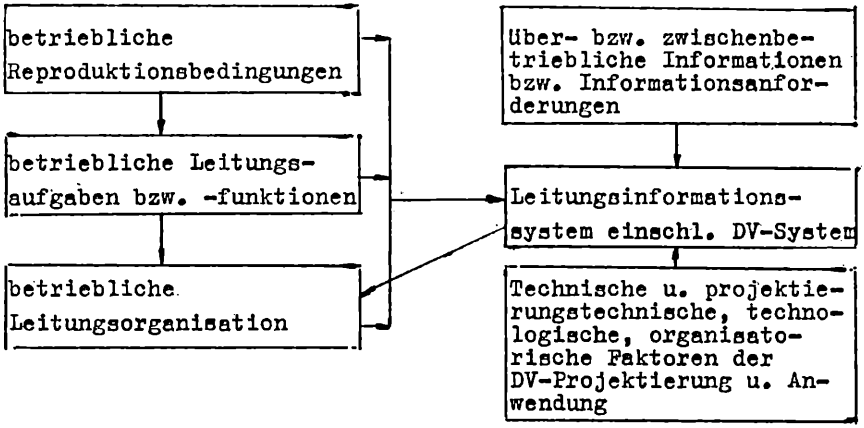
Einleuchtend erscheint, daß das Phänomen Mehrfachnutzung primär von 2 Faktoren abhängig ist /33/:

1. von bestimmten Eigenschaften der projektierten und mehrfach zu nutzenden DV-Lösung,
2. von den konkreten Anwendungsbedingungen einer DV-Lösung.

Dabei ist trivial, daß bei einheitlichen Anwendungsbedingungen, wenn man darunter sowohl die inhaltliche Seite als auch die technischen und organisatorischen Bedingungen versteht, einheitliche DV-Lösungen angewandt werden können und damit die Mehrfachnutzung ihr Maximum erreichen kann /33/. Die Grenze einer solchen maximalen Nachnutzung für problembezogene DV-Lösungen liegt im Leitungsbereich in den sogenannten "betrieblichen Reproduktionsbedingungen", worunter man nach TRAGSDORF /39/ volkswirtschaftliche, wirtschaftsorganisatorische, technologische und personelle Bedingungen versteht. Den für die Gestaltung eines Leitungsinformationssystems von TRAGSDORF dargestellten Wirkungszusammenhang zeigt folgende Abbildung /39/:



Diese Darstellung charakterisiert die bestehenden prinzipiellen Abhängigkeiten eines betrieblichen Informationssystems m. E. unvollständig und damit zu einseitig, wie anhand der folgenden Folie gezeigt werden soll /33/:




Die zuletzt dargestellten Zusammenhänge zeigen m. E. die richtige Einordnung eines betrieblichen Informationssystems als Verbindungsglied bzw. Kommunikationsmittel zwischen den genannten Einflußfaktoren, da alle Leitungsbeziehungen ausschließlich Informationsbeziehungen darstellen /33/.

Dabei besteht der Zusammenhang, daß bei gleichen Reproduktionsbedingungen ein einheitliches Informationssystem realisierbar ist, jedoch gilt nicht die Umkehrung, daß bei unterschiedlichen betrieblichen Reproduktionsbedingungen keine einheitlichen Leitungsinformationen sinnvoll sind.
Daraus leiten sich folgende prinzipielle Möglichkeiten der Intensivierung ab:

a) die Herausarbeitung solcher Informationsverarbeitungsprozesse, die trotz unterschiedlicher betrieblicher Reproduktionsbedingungen rationell mit Hilfe der EDV einheitlich gestaltet werden können,

b) eine solche Organisation der DV-Projektierung, die jeweils solche Betriebe in die Mehrfachnutzung arbeitsteilig zu projektierender DV-Lösungen einbezieht, die eine hohe Übereinstimmung bzw. geringe Abweichungen der betrieblichen Reproduktionsbedingungen aufweisen.

c) die Anwendung solcher Methoden und Technologien der DV-Projektierung, die trotz bestimmter Abweichungen in den betrieblichen Reproduktionsbedingungen eine rationelle Mehrfachnutzung gestatten. /33/ /30/

Aus diesen Lösungsansätzen gehen wesentliche spezifische Anforderungen an die Wege zur Intensivierung des spezifischen Arbeitsprozesses "DV-Projektierung" hervor, auf die ich im folgenden unter dem Aspekt der Vereinheitlichung von DV-Lösungen eingehen will.

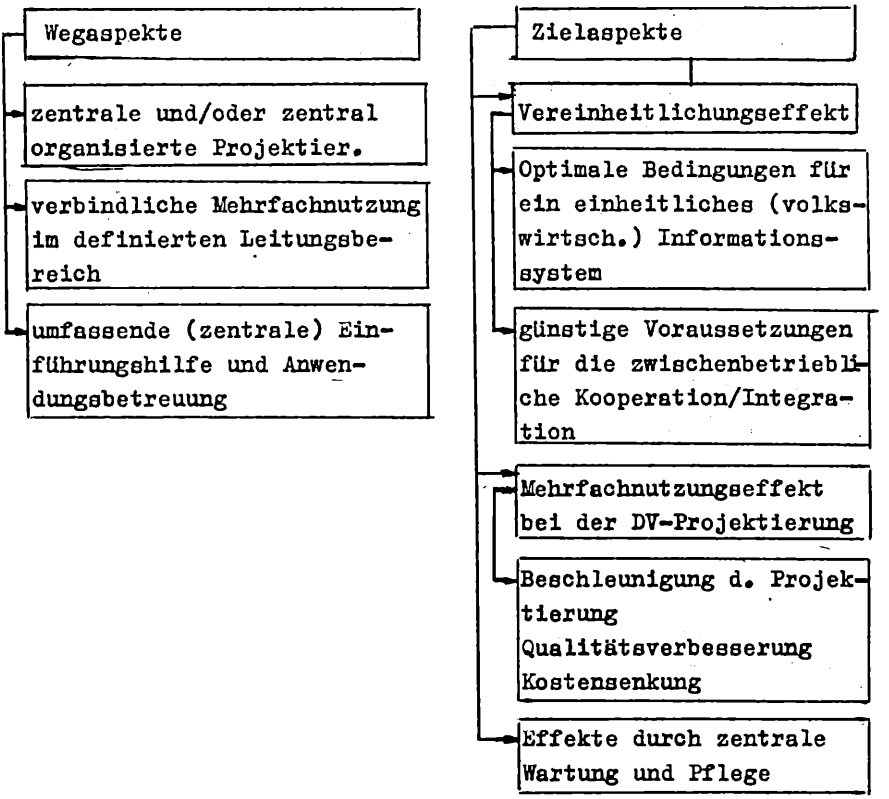
## 2.5. Vereinheitlichung und variable Gestaltung von DV-Lösungen als Wege zur Schaffung mehrfach nutzbarer DV-Lösungen

Die Frage, welcher Weg, Vereinheitlichung oder Variabilität gegangen werden soll, kann nicht für alle DV-Aufgaben und alle Betriebe gleich beantwortet werden.
Bereits bei der Darlegung der von mir vertretenen Hauptwege der Rationalisierung der DV-Projektierung sind 3 praktikable Hauptformen der Projektgestaltung genannt worden:

Die Projektierung von

- Einheitsprojekten
- teilweise variablen DV-Lösungen und
- Variablen DV-Lösungen. /33/

Die folgende Übersicht enthält die wichtigsten Weg- und Zielaspekte für Einheitsprojekte.

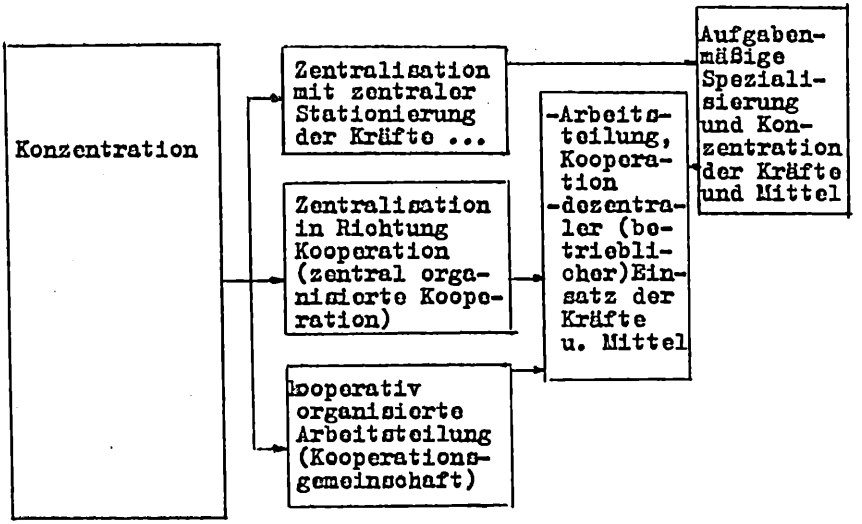



Bei den anderen beiden Formen werden diese Ziele mit anderen Methoden verfolgt, auf die ich hier im einzelnen nicht eingehen kann. /33/

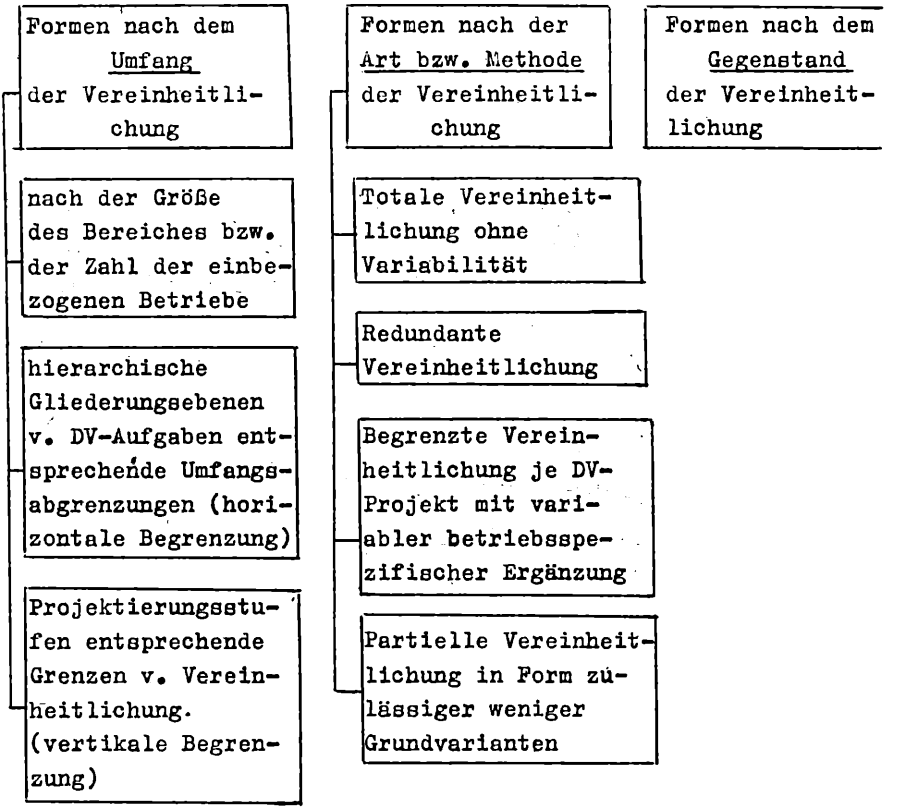
Vereinheitlichungen von DV-Projekten sind an bestimmte überbetriebliche bzw. gesellschaftliche Organisationsformen gebunden, auf die im folgenden eingegangen werden soll.

## 2.6. Rationalisierung der DV-Projektierung durch Formen der gesellschaftlichen Organisation

Aus folgender Übersicht gehen die prinzipiellen Formen der gesellschaftlichen Organisation bei der DV-Projektierung hervor.



In der folgenden Übersicht sind theoretisch mögliche Vereinheitlichungsformen für DV-Projekte enthalten, deren Anwendungsbedingungen für mehrere Kooperationsbereiche untersucht wurden. /33/




In der Praxis der kooperativen Projektierung haben sich Formen einer annähernd totalen Vereinheitlichung in Form von Einheitsprojekten für einen Industriebereich sowie besonders Formen der redundanten Vereinheitlichung bewährt. Aus der Vordruckvereinheitlichung sind die beiden anderen Formen bekannt geworden. Diese haben jedoch für eine EDV-Kooperationsgemeinschaft geringe Bedeutung. /33/

Es ist zu sehen, daß sich die Konzentration nur in Form der Zentralisation und/oder Kooperation realisieren läßt und daß diese Formen letztlich zu einer aufgabenmäßigen Spezialisierung und Konzentration der Kräfte und Mittel führen.
Die Konzentration ermöglicht neben der aufgabenmäßigen bzw. "thematischen" Konzentration und Spezialisierung wesentliche Vorteile des von Marx herausgearbeiteten einfachen Konzentrationseffektes, wenn mehr Projektanten in einem Gebäude stationiert werden, hat jedoch wesentliche Grenzen in den materiellen und personellen Voraussetzungen. /33/
Die Zentralprojektierung bietet günstige Möglichkeiten zur Realisierung der Konzentration der DV-Projektierung in einem Industriebereich oder Kombinat, insbesondere mit dem Ziel der Entwicklung und Anwendung, einschließlich Wartung und Pflege von Einheitsprojekten sowie Querschnittslösungen und der Bearbeitung von Schwerpunktaufgaben des gesamten Bereiches. /33/
Die Bedeutung der zwischenbetrieblichen Kooperation ergibt sich einerseits aus den Grenzen der Zentralisation und Konzentration, u. a. der materiellen Realisierbarkeit, andererseits aus der Möglichkeit, besonders die aufgabenmäßige Konzentration und Spezialisierung ohne materielle Umwälzungen zu realisieren. /31/

Die Kooperation kann als gegenwärtig wichtigste Form der gesellschaftlichen Organisation der DV-Projektierung in der Industrie der DDR bezeichnet werden. /33/
Die Organisation der zentral oder kooperativ organisierten arbeitsteiligen DV-Projektierung in den neu gebildeten Kombinaten sollte unbedingt auf den bisherigen umfangreichen Erfahrungen und wissenschaftlichen Untersuchungen aufbauen./33/

Ich möchte heute einige Anregungen zu diesen Fragen aus der Sicht des in den nächsten Jahren zu erwartenden massenhaften Einsatzes von Bürocomputern geben.

## 2.7. Fragen der gesellschaftlichen Organisation der DV-Projektierung unter der Bedingung des Einsatzes von Bürocomputern

Neu stellt sich gegenwärtig, mit dem Beginn des massenhaften Einsatzes von Bürocomputern, die Frage der Bereitstellung der DV-Projektierungskapazität.

Da hiervon tendenziell Betriebe bis herunter zur Betriebsgröße von weniger als 20 Beschäftigten betroffen werden, besteht das Problem, daß solche Betriebe kaum über einen EDV-Spezialisten verfügen, obwohl die Programmierung von Bürocomputern kaum geringere Schwierigkeiten aufweist, als die ESER-Programmierung. In diesem Zusammenhang sind m. E. die Fragen einer Zentralprojektierung auf der Ebene der Kombinate und Ministerien sowohl in Form von Einheitsprojekten (einschließlich Muster-Organisation) als auch von variablen DV-Lösungen neu zu durchdenken. U. U. erweisen sich hierzu auch spezielle Abteilungen des Kombinates DV sowie evtl. neu zu schaffende territoriale Beratungs- und Softwarezentren, auf Dienstleistungsbasis und auf der Grundlage vertraglicher Bindungen als geeignete gesellschaftliche Organisationsform.

Aufgrund der hohen Zahl an möglichen Nachnutzungen von Soft- und Orgware wäre hier ein auch fachlich und vom erzielbaren Nutzen her interessantes Betätigungsfeld für hochqualifizierte Spezialisten als Org- und Software-Berater und -Entwickler denkbar. Dadurch könnte das m. E. vorhandene diesbezügliche Vakuum bei den Kleinbetrieben volkswirtschaftlich mit geringstem Arbeitsaufwand gemindert werden.
Eine ausschließlich oder weitgehend betriebliche Einsatzvorbereitung kann m. E. auf keinen Fall für eine Betriebsgröße bis zu wenigen 100 Beschäftigten vertreten werden.
Auf die von mir skizzierte Weise wäre auch eine u. U. jahrzehntelange thematische Spezialisierung auf einzelne Sachgebiete, z. B. ein "Informationssystem für Material- und Lagerwirtschaft" möglich, wodurch ein normaler effektiver Lernprozeß aus den vorangegangenen Einsatzvorbereitungsfällen mög-

lich wird. Wenn man dazu noch die Spezialisten für ein Sachgebiet strukturell und lokal zusammenfaßt, wird allein durch diesen thematischen Konzentrationseffekt der eindeutige Vorteil für den Auftraggeber, u. U. auch für einen Exportkunden deutlich. Analog sollte die Frage der Organisation gesehen werden: ein solches Organisations- und Software-Beratungsbüro sollte mit den so stets vorhandenen Erfahrungen in der Lage sein, einen Kleinbetrieb in kurzer Zeit relativ komplex mit Mitteln der Organisation und IV-Technik zu rationalisieren, die Belegschaft zu schulen und sich dem nächsten Betrieb im Territorium bzw. Kombinat zuzuwenden.
Dazu muß jedoch eine neue Kooperationskette zwischen
- Hersteller der Hardware,
- Hersteller der Grundsoftware,
- Org- und Software-Beratungsbüro (als gleichzeitigem Applikationssoftware-Entwickler) und
- Anwenderbetrieb

hergestellt werden. Durch diese Kette mußdie Bereitstellung der Hard- und Software spätestens dann gesichert werden, wenn dies im Vertrag mit dem Anwenderbetrieb vereinbart wurde, so daß das Beratungsbüro auch die Funktionsfähigkeit und den Nutzen des neuen Organisationssystems nachweisen kann.

3. Zum Verhältnis von Organisations- und DV-Projektierung- Stellungnahme zum "Leitfaden Organisationsprojektierung" /3/

Das Anliegen des Autorenkollektivs, mit dem Leitfaden einen Beitrag zu planmäßiger und systematischer Organisationsarbeit zu leisten, muß auch von allen Fachkollegen, die sich mit der DV-Projektierung beschäftigen, unterstützt werden. Ich möchte an dieser Stelle nur auf meinen Haupteinwand eingehen: Dieser besteht darin, daß die DV-Projektierung als Teilgebiet der Organisationsprojektierung aufgefaßt wird und phasenverschoben nach den ersten sogenannten O-Stufen beginnt.

Das bedeutet, daß die Aufgabenstellungen für alle DV- Anwendungen, letztlich auch ein komplexes ASU, vor der DV-Projektierung konzipiert und als Vorgabe an die EDV-Organisation (vgl. S. 47, S. 64) übergeben werden, wobei lt. Leitfaden u. a. folgende Teillösungen der DV-Projektierung vorgegeben werden sollen:

- ein "Makroalgorithmus",
- ein "Mikroalgorithmus",
- weitere bisher in der DV-Projektierung in den Stufen E1 bis E3 gelöste Aufgaben, wie Schlüssel- und Vordruckgestaltung.

Dies bedeutet:

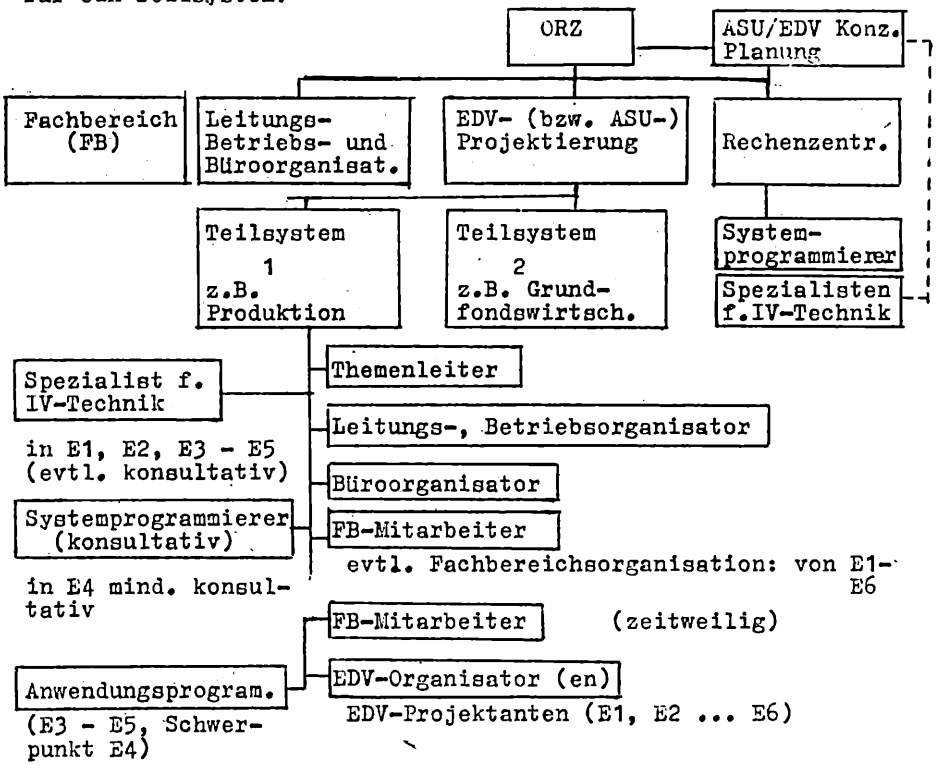
a) einen sehr hohen Bedarf an solchen universellen Orgspezialisten
b) ein späteres Eintakten der EDV-Organisatoren in den Projektierungsprozeß und deshalb lange Einarbeitungszeit oder geringe Problemkenntnisse der EDV-Organisatoren, d. h. Nachteile bei der DV-Projektierung.

Dabei ergibt sich die prinzipielle Frage, ob ein Nicht-EDV-Spezialist die Frage der Automatisierbarkeit und Automatisierungswürdigkeit a priori besser lösen kann als ein EDV-Organisator während der DV-Projektierung? Dagegen sprechen einige Argumente, u. a.:

a) die komplizierter werdende Informationsverarbeitungs-Gerätetechnik,
b) der steigende Anteil der Software am Gesamtaufwand. Die Automatisierungswürdigkeit /41/ hängt damit primär von der Existenz und Anwendbarkeit bzw. rationellen Möglichkeit der Eigenentwicklung geeigneter Grund- und Applikationssoftware ab.

Deshalb sollten meiner Ansicht nach die Fragen der Automatisierungswürdigkeit durch ein interdisziplinäres Projektantenkollektiv beantwortet werden.

Meines Erachtens sind Unterstützungsaktivitäten der Leitungs- und Büroorganisatoren zweckmäßiger als eine zeitliche Trennung dieser Organisation von der DV-Organisation. Ein solches Vorgehen ist m. E. ein Grundgedanke der ASU-Projektierung /1/, bei der ASU-Teilsysteme durch ein ASU-Projektantenkollektiv aus Spezialisten mehrerer Disziplinen simultan und komplex bearbeitet werden.
Daraus ergibt sich folgender Vorschlag zur Einbeziehung unterschiedlicher Spezialisten in die ASU- bzw. EDV-Projektierung für ein Teilsystem:



Einige Schlußfolgerungen:

Aus meinen Darlegungen leitet sich die These ab:
DV-Projektierung muß stets gleichzeitig als Org-Projektierung aufgefaßt werden.

Die ersten Projektierungsstufen sollten neben der stets gebotenen Einbeziehung der betreffenden Fachbereiche eine Gemeinschaftsarbeit von Leitungs- sowie Büro- und EDV-Organisatoren sein. Dabei kann man dort, wo der EDV-Einsatz zeitweilig nicht vordergründig gesehen werden kann, zunächst durchaus von "Organisationsprojektierung" reden. Die O-Stufen sollten jedoch nur für die Teile des Projektierungsgegenstandes weitergeführt werden, für die kein EDV-Projekt erarbeitet wird. Daraus ergibt sich als weitere Folgerung:
Die Dokumentationen der ersten O- und E-Stufen sollten soweit wie möglich übereinstimmen, um Doppelarbeiten zu vermeiden.
Deshalb sollte von EDV- und Organisationsspezialisten gemeinsam an der weiteren Präzisierung des Leitfadens zur Organisationsprojektierung für den genannten Anwendungsbereich gearbeitet werden.

## 4. Organisationsarbeit als Objekt der EDV-Anwendung (rechnergestützte Organisationsarbeit)

Organisationsarbeit als überwiegend geistig-schöpferische weitgehend entwerfende, planende, gestaltende Tätigkeit mit diesbezüglichen Parallelen zur DV-Projektierung muß wie jede gesellschaftliche Arbeit selbst rationell gestaltet werden. Aus meiner Sicht ergeben sich folgende Schwerpunkte einer künftigen Rechnerstützung für die Organisatorentätigkeit:

1. Die umfassende Anwendung von universellen Textbe- und -verarbeitungssystemen als erster und vielleicht schon für die nächsten 5 - 6 Jahre wichtigster Schritt, um Org- und Arbeitsanweisungen und organisatorische Regelungen rationell formulieren und warten zu können.

2. Die Schaffung spezieller Verwaltungs- und Kontrollsysteme für Dokumentationen von organisatorischen Lösungen.

3. Die Entwicklung eigens für die Organisationsarbeit zu schaffender Fachsprachen und geeigneter Software für die Formulierung organisatorischer Sachverhalte und die erforderliche synthaktische und teilweise semantische Prüfung.

4. Die Anwendung der grafischen Datenverarbeitung für die Darstellung von Prozessen und Strukturen.

5. Die Formalisierung typischer organisatorischer Fragestellungen und von Algorithmen für rechnergestützte Entscheidungshilfen bei der Organisationsarbeit.

6. Die umfassende Nutzung der betrieblichen DV-Systeme, z. B. von Datenbanksystemen, auch zur Vorbereitung organisatorischer Entscheidungen.

Dabei können m. E. viele Hilfsmittel, die für die rechnergestützte DV-Projektierung genutzt werden, auch für die Leitungs- und Büroorganisation angewandt werden.

## 5. Einige methodologische Zusammenhänge zwischen Organisation und DV-Projektierung und daraus resultierende Schlußfolgerungen

Begriffe wie Nutzerakzeptanz, Nutzerfreundlichkeit und "Mensch-Maschine-Symbiose" /17/ weisen seit Ende der 70er Jahre auf ein neu entstandenes Problem in der Automatisierung der Informationsverarbeitung hin: auf das Problem der Mensch-Maschine-Kommunikation, wobei die Arbeit des Menschen, sein Arbeitsprozeß als primär anzusehen ist. /29/
Die von GRIESE und ÖSTERLE /17/ vertretene neue Spezialisierung in Form eines Requirements Engeneering sollte uns in diesem Zusammenhang nicht unbedingt darauf hinweisen, eine neue Ingenieurdisziplin für die Fragen der Arbeitsorganisation und Arbeitsbedingungen im Zusammenhang mit der Dialogverarbeitung zu schaffen, sondern die EDV-Organisatoren bzw. DV-Projektanten auf diese Aspekte stärker zu orientieren. /29/

Um die Frage der Spezialisierung innerhalb der DV-Projektierung in Anwenderbereichen aus meiner persönlichen Sicht zu beantworten, möchte ich hier bewußt zwischen zwei Extremen unterscheiden:

- einerseits großen betrieblichen zentralen oder territorialen Projektierungskollektiven und i. d. R. größeren Projektkomplexen und Betrieben als Gegenstand und größeren EDVA als Mittel und
- andererseits sehr kleinen Projektierungskollektiven für Kleinbetriebe mit dem primären Mittel Bürocomputer.

Für den ersten Fall habe ich zur Entwicklung organisatorischer Spezialdisziplinen bereits Aussagen gemacht.
Auf der Seite der DV-Spezialisten, also der Angewandten Informationsverarbeitung, sehe ich primär eine Spezialisierung in Form folgender Disziplinen als sinnvoll an:

1. EDV-Organisatoren (Organisationskenntnisse, Kenntnisse auf dem Fachgebiet, Kenntnisse der Anwendungsprogrammierung),
2. Anwendungsprogrammierer bzw. Applikations-Software-Ingenieure (Kenntnisse auf dem Gebiet des Software-Ingenieurwesens, Kenntnisse auf dem Fachgebiet, geringe Kenntnisse auf organisatorischem Gebiet),
3. System-Software-Ingenieure (Systemprogrammierer, Spezialisten für Entwurf, Implementierung und Test großer Programmsysteme),
4. DV-Projektanten (Spezialisierungsrichtung mit Kenntnissen der EDV-Organisation und der Anwendungsprogrammierung (Software-Entwurf, -implementierung und Test) i. d. R. in höheren Programmiersprachen).

Der Anteil der einzelnen Disziplinen sollte je nach Projektierungsobjekt differenziert werden. Meiner Ansicht nach braucht lediglich der unter 3. genannte Software-Ingenieur bzw. Systemprogrammierer eine Ausbildung in Informationsverarbeitung, wie sie in den beiden Dresdener Sektionen erfolgt, für die 2. Kategorie erscheint deren Einsatz möglich. Die anderen Disziplinen können mit einem Fachstudium auf dem jeweiligen Anwendungsgebiet nach den Richtlinien der im RGW abgestimmten ASU-Kadergruppe IVa ausgebildet werden, wobei ein IV-Anteil von 25 % des Studiums anzustreben ist.
Bei EDV-Organisatoren halte ich eine von vornherein organisationswissenschaftlich orientierte Ausbildung, ähnlich der an der Humboldt-Universität existierenden für Wissenschaftsorganisatoren, für zweckmäßig.
Für große Bereiche der Technischen Vorbereitung, z. B. große Konstruktions- und Projektierungsbüros, erscheinen solche Wissenschaftsorganisatoren mit einer IV-Vertiefung /40/ aus meiner Sicht besonders geeignet.
Für Kleinbetriebe, besonders für den Einsatz von Bürocomputern in Kleinbetrieben, erscheint der relativ universell einsetzbare DV-Projektant, unterstützt von einem Büroorganisator, die zweckmäßigste Spezialisierungsform.

Geht man von der Notwendigkeit der Realisierung der DV-Projektierung als interdisziplinäre Aufgabe aus, da sonst i. d. R. entweder schlechte Software- oder nicht befriedigende Orgware entsteht /17/, ergibt sich, daß für diese interdisziplinäre Arbeit auch eine einheitliche Technologie und eine einheitliche Rechnerstützung erforderlich ist. Da DV-Projektierung in den meisten Applikationen nur schätzungsweise 40 - 60 % Software-Ingenieur-Arbeit, ansonsten aber Organisationsarbeit darstellt, erweisen sich Versuche, die gleichen Technologien des Software-Ingenieurwesens, wie sie bei reinen Software-Produzenten notwendig sind, unvermittelt und ungefiltert den Anwenderbetrieben anzubieten, als problematisch. Dies zeigt sich sehr deutlich in Versuchen, z. B. unterschiedliche Programmentwicklungsmethoden, wie die Jackson-Methode, funktionsorientiertes Entwerfen und andere möglichst parallel dem gleichen Projektanten zur selbständigen Auswahl anzubieten, Verwaltungssysteme für Aufgabenstellungen, überhaupt Aufgabenstellungen aus der Rechnerstützung jedoch auszuklammern. Spezielle für die Organisation geeignete Entwurfsmethoden, wie die SADT-Methode, wurden aus der bisherigen Diskussion meist von vornherein ausgeklammert. Wenn der DV-Projektant auch Organisator sein soll, muß ihm auch zugestanden werden, nicht das Software-Entwicklungswissen eines spezialisierten Software-Ingenieurs zu haben. Aus dieser Sicht müssen software-technologische Fragen in engerem Zusammenhang zu orgware-technologischen im Prozeß der DV-Projektierung betrachtet werden. Daraus ergibt sich zunächst einmal die Forderung an die Organisationswissenschaft, aber besonders an die wissenschaftliche Methodologie der DV-Projektierung, diese, die EDV-Organisation betreffenden Teile der DV-Projektierung im Sinne einer Technologie, die die Aspekte der Rechnerstützung von vornherein einbezieht, weiterzuentwickeln. Wichtig erscheint dabei, daß die verwendeten Instrumentarien von der Organisationsanalyse bis hin zum Programmentwurf und zur Projektdokumentation aufeinander abgestimmt sind und einen nahtlosen Übergang zwischen den einzelnen Bearbeitungsstufen ermöglichen.

In diesem Sinne sind sowohl die existierende Rahmenmethodik der DV-Projektierung als auch entwickelte softwaretechnologische Ansätze neu zu durchdenken, da DV-Projektierung sich eben nicht auf das Software-Ingenieurwesen beschränken läßt, sondern eine immer bedeutendere Orgware-Komponente haben wird.

Verfasser: Prof.Dr.sc. Köhli, Siegfried
Ingenieurhochschule Köthen
Abt. Mathematik/Rechentechnik
WB Informationsverarbeitung
4370 Köthen
Bernburger-Str.52-57

Literaturverzeichnis

1 Afanasjew, W.G.:
Soziale Information und Leitung der Gesellschaft
Staatsverlag der DDR Berlin 1976

2 Autorenkollektiv:
Informatik und Automatisierung Band 1
Theorie und Praxis der Struktur und Organisation der Informationsverarbeitung
Akademie – Verlag Berlin 1976

3 Autorenkollektiv:
Leitfaden zur Organisationsprojektierung
Verlag Die Wirtschaft Berlin 1981

4 Autorenkollektiv:
Organisationstechnik
fuer Leitungs- und Verwaltungsarbeiten
Staatsverlag der DDR Berlin 1982

5 Autorenkollektiv:
Oekonomisches Lexikon, Band H – P.
3.Auflage. Verlag Die Wirtschaft Berlin 1979,
S.645

6 Autorenkollektiv:
Organisation in Leitung und Verwaltung.
Methodische Konzeptionen und Erfahrungen der Rationalisierung
Staatsverlag der DDR Berlin 1974

7 Bannester, E.M.:
Complex Organization and Complex of Organizations
Twente 1971. Dissertation

8 Beck, H.:
Was ist Organisation? Wer ist Organisator?
In: Zeitschrift fuer Organisation (ZfO) 8(1934)9,
S.368–369

9 Belke, W.:
Die Informationsverarbeitung als technologische Wissenschaftsdisziplin.
Dissertation B. Technische Universitaet Dresden,
Sektion Informationsverarbeitung
Dresden 1979

10 Daeumichen, K.:
Leitungsorganisation in Wissenschaft und Forschung – Probleme, Erfahrungen, Tendenzen.
In: Leitung der Hochschulforschung – Ausgewaehlte Probleme und Erfahrungen.
Institut fuer Hochschulforschung, Berlin 1981,
S.108–126

11 Daeumichen, K.:
Zum Problem der Bewertung oekonomischer Informationen
bei der Vorbereitung von Leitungsentscheidungen.
In: Wirtschaftswissenschaft 16(1968)1, S.77-86

12 Dobrov, G.M.:
Organisationstechnologie als Gegenstand der Systemanalyse.
Teil 1: Grundsatzfragen
In: Wissenschaftliche Zeitschrift der Humboldt - Universitaet
zu Berlin, Math.-Nat.R. XXVIII
(1979)5; S.613 ff

13 Dobrov, G.M.:
Organisationstechnologie als Gegenstand der Systemanalyse
Teil 2 : Aspekte und Ebenen der Organisation.
In: Wissenschaftliche Zeitschrift der Humboldt - Universitaet
zu Berlin, Math.-Nat.R. XXVIII (1979)5;
S.675 ff

14 Dyllik, Th.:
Entscheidungsorientierte und systemorientierte Ansaetze
In: Menagement - Zeitschrift 50(1981)12
Verlag Industrielle Organisation BW/ETH
Zuerich

15 Fuchs-Kittowski, K.; Kaiser, H.; Tschirschwitz, R.;
Wenzlaff, B.:
Informationen als Verhaeltnis zwischen physikalischer
Wirkung und Organisation
In: Wissenschaftliche Zeitschrift der Humboldt - Universitaet
zu Berlin, Ges.Sprachw.R. XXVI (1977)5,
S.627 ff

16 Garczarczyk, J.; Adamczewski, P.:
Rechnergestuetzte Datenverarbeitungsprojektierung
auf der Basis PSL/PSA-Pakets
In: Rationalisierung der Datenverarbeitungsprojektierung
Kongress - und Tagungsberichte der
Martin - Luther - Universitaet Halle - Wittenberg.
Wissenschaftliche Beitraege 1981/23 (DZ7),
Halle/S. 1981

17 Griese, J.; Oesterle, H.:
Requirements Engineering
In: Angewandte Informatik, Nr. 4/1978,
S.150 ff

18 Grochla, E.(Hrsg.):
Das Buero als Zentrum der Informationsverarbeitung
In: Aktuelle Beitraege zur Betriebswirtschaftlichen
Forschung. Wiesbaden 1971

19 Grochla, E.(Hrsg.):
Handwoerterbuch der Organisation.
Stuttgart 1973

20 Grochla, E.:
Integrierte Gesamtmodelle der Datenverarbeitung
Entwicklung und Anwendung des Koellner Integrationsmodells
(KIM).
Muenchen 1974.

21 Grochla, E.; Foerster, G.:
Organisationsplanung und Organisationsentwicklung – Theorie und Praxis
In: Schriften der Arbeitsgemeinschaft fuer Rationalisierung des Landes Nordrhein-Westfalen.
Dortmund 1977

22 Grochla, E.:
Organisationstheoretische Ansaetze zur Gestaltung rechnergestuetzte Informationssysteme
In: Angewandte Informatik, Nr.4/1978; S.141

23 Grochla, E.:
Unternehmensorganisation. Neue Ansaetze und Konzeptionen.
Reinbeck bei Hamburg 1972

24 Hofmann, F.:
Entwicklung der Organisationsforschung.
Betriebswirtschaftlicher Verlag Dr. Th. Gabler
Wiesbaden 1973

25 Hoffmann, M.J.A.:
Betriebliche Informationswirtschaft und Datenverarbeitungsorganisation.
Analyse und Konzeption von Organisationssystemen.
Berlin (West), New York 1976

26 Honnecker, E. (Berichterstatter):
Bericht des Zentralkomitees der Sozialistischen Einheitspartei Deutschlands an den X.Parteitag der SED
Dietz Verlag Berlin 1981

27 Kacir, K.:
Auf der Nutzung der Rechentechnik basierende Systeme der Leitung (russ).
Autorreferat zur Erlangung des akademischen Grades Dr.sc.oec.
Bratislava, Juli 1980

28 Kirsch, W.; Meffert, H.:
Organisationstheorien und Betriebswirtschaftslehre.
In: Schriftenreihe der Zeitschrift fuer Betriebswirtschaft, Bd.1.
Wiesbaden 1970

29 Klatt, G.:
Organisatorische Aspekte der Bewertung automatisierter Informationssysteme
In: Rationalisierung der Datenverarbeitungsprojektierung.
Wissenschaftliche Beitraege der Martin – Luther – Universitaet Halle – Wittenberg 1981/23(D 27)

30 Koehli, S.:
Hauptwege der Intensivierung der Datenverarbeitungs-projektierung.
In: Rechentechnik/ Datenverarbeitung
17(1980)1, S.29 f

31 Koehli, S.:
Organisationsformen der Kooperation bei der Intensivierung der DV-Projektierung
In: Rechentechnik / Datenverarbeitung 18(1981)5,
S.31 ff

32 Koehli, S.:
Projektierungsvorgaben zur Durchsetzung einer rationellen DV - Projektierung
In: Rechentechnik / Datenverarbeitung 18(1981)7,
S.30 ff

33 Koehli, S.:
Intensivierung der Datenverarbeitungsprojektierung durch Kooperation.
Dissertation B. Technische Hochschule ''Carl Schorlemmer''
Leuna - Merseburg 1978

34 Koehli, S.:
Rationalisierung des Informationsaustausches
In: Rechentechnik / Datenverarbeitung 15(1978)10,
S.5 ff

35 Luhmann, N.:
Organisation und Entscheidung.
Rheinisch - Westfaelische Akademie der Wissenschaften.
Westdeutcher Verlag GmbH Opladen 1978

36 Neumann, J.; Schneider, H.; Uhlmann, Th.:
Komplexe Durchfuehrung von Aufgaben der Leitungsorganisation in Betrieben und Kombinaten.
In: Wissenschaftliche Zeitschrift der Technischen Hochschule ''Carl Schorlemmer''
Leuna - Merseburg 18(1976)4

37 Schneider, H.; Koehli, S.:
Systematische Projektierung der Datenverarbeitung
Verlag Die Wirtschaft Berlin 1980

38 Taylor, F.W.:
Die Grundsaetze wissenschaftlicher Betriebsfuehrung.
Muenchen, Berlin 1973

39 Tragsdorf, K.:
Zur Erfassung und Analyse des betrieblichen Leitungsaufbaus
In: Autorenkollektiv: Leitungsorganisation in den Betrieben und Kombinaten.
Schriften zur sozialistischen Wirtschaftsfuehrung
Dietz Verlag Berlin 1980, S.174

40 Tschirschwitz, R.:
Organisationstaetigkeit aus der Sicht der Gestaltung von Informationssystemen und einige Aspekte der Aus- und Weiterbildung von Wissenschaftsorganisatoren.
Ungedrucktes Manuskript.
Humboldt - Universitaet zu Berlin 1982

41 Tschirschwitz, R.:
Wissenschaftstheoretische und Methodologische Aspekte der Informatik - Zum Verhaeltnis von Information und Organisation sowie der Rolle von Informationstechnologien in der Gesellschaft
Dissertation B. Humboldt- Universitaet zu Berlin 1981

42 Tschirschwitz, R.:
Zur Analyse des Verhaeltnisses von Information und Organisation als Veraussetzung der Gestaltung von Informationssystemen.
Vortrag auf dem Kolloquium ''Rationalisierung der DV-Projektierung''
Martin - Luther - Universitaet Halle - Wittenberg
Halle,S. Im Oktober 1980
(Thesen in : /28/, S.30-33)

43 Yourdon, E.:
Umwaelzungen auf dem Gebiet der Softwareentwicklung.
In: GMD - Spiegel 8(1978)2, S.5-21

44 VEB Kombinat Robotron:
Aktualisierung und Ergaenzung der Rahmenmethodik der Datenverarbeitungsprojektierung vom Maerz 1973.
Dresden 1976

Dipl.-Math. Hans Winks, Humboldt-Universität

# SPOOL - ein System zur Automatisierung von Produktionsprozessen an ESER-Rechnern - Entwicklungsstand und Entwicklungsrichtungen -

## 1. Vorbemerkungen

Die EDV-Anlagen, die zur Zeit in der DDR eingesetzt werden, haben eine hohe Stufe der Leistungsfähigkeit erreicht. Insbesondere die Anlagen des ESER I und des ESER II prägen in vielen Großrechenzentren Arbeitsorganisation, Projektierungs- und Programmierungsstil, wobei z.B. auf dem Gebiet der Rationalisierung datenintensiver Prozesse in den letzten Jahren große Fortschritte zu verzeichnen sind.

Auf diesen Anlagen wird zur Zeit überwiegend das Betriebssystem OS/ES mit den Steuerprogrammkonfigurationen MVT und MFT eingesetzt. Dieses Betriebssystem bietet sowohl den Nutzern als auch den Anlagenbedienern umfangreiche Möglichkeiten zur Steuerung der Jobabarbeitung in der EDVA und hat sich deshalb als das bisher leistungsfähigste Betriebssystem für die mindestens bis 1990 eingesetzten ESER-Anlagen erwiesen. Trotz des hohen Leistungsgrades der ESER-EDVA wird vielerorts daran gearbeitet, durch noch bessere Auslastung der Ressourcen einen Produktivitätszuwachs zu erreichen. Dabei geben die Beschlüsse und Orientierungen des X. Parteitages der SED und des 10. Kongresses des FDGB die Richtung an, nämlich durch Intensivierung der Nutzung der hochproduktiven Anlagen und Investitionsobjekte eine Steigerung der Arbeitsproduktivität zu erreichen.

Einerseits durch die Vielfalt und die Komplexität der teilweise eine recht detaillierte Systemkenntnis voraussetzenden Möglichkeiten des OS/ES, andererseits durch eine mangelnde oder fehlende Unterstützung wesentlicher Prozesse im Rechenbetrieb außerhalb der unmittelbaren Jobverarbeitung in der EDVA und nicht zuletzt durch eine unzureichende Orientierung auf Vereinfachung, Produktivitätssteigerung und Automatisierung der Abläufe im Umkreis der EDVA ergeben sich Grenzen bezüglich der Erhöhung der Gesamteffektivität und damit Ansätze für die Entwicklung zusätzlicher Systemsoftware. Es seien an dieser Stelle einige Probleme genannt, die aus Effektivitäts- und Zuverlässigkeitsgründen einer rechnergestützten Lösung bedürfen.

I. Für die Organisation eines reibungslosen Stapelbetriebes ist ein beträchtlicher Arbeitsaufwand in der Arbeitsvor- und -nachbereitung nötig. Insbesondere betrifft das die während der Jobvorbereitung vorzunehmende Auswertung der üblichen Jobbegleitkarten, die notwendig ist
   - für die Zusammenstellung der Jobs nach bestimmten, für die Abarbeitung vorteilhaften Gesichtspunkten (z.B. Datenträgernutzung),
   - für die Prüfung der Berechtigung der Nutzung der verschiedenen Rechnerressourcen (z.B. Zeit, Datenträger, Dateien) durch die Jobs.

II. Auf Grund der erwähnten Komplexität des OS/ES ist der Bedienaufwand und der Einfluß der Bediener auf die Gesamtproduktivität der Anlage recht groß, insbesondere dann, wenn Aufgaben der Arbeitsvorbereitung vom Bedienpersonal wahrgenommen werden. Der Arbeitsablauf und die Anforderungen der Nutzerjobs bzw. des Betriebssystems an die Bediener erfordern deren ständige Aufmerksamkeit. Das OS/ES bietet von sich aus kaum Möglichkeiten, durch eine geeignete Jobauswahl die Abarbeitung des Jobstromes so zu gestalten, daß Bedienhandlungen minimiert werden (z.B. Datenträgerwechsel) bzw. daß Bedienhandlungen auf bestimmte Tageszeiträume konzentriert werden, um das Bedienpersonal sowohl zahlenmäßig als auch bezüglich der Arbeitsintensität effektiver einsetzen zu können.

III. Die Anlagenbedienung weist unter OS/ES einige Unbequemlichkeiten und Schwächen auf. Das betrifft zum Beispiel:
   - die Informationsmöglichkeiten für die Bediener über die Jobs (Jobbegleitkarte),
   - das zeitaufwendige Realisieren von Maßnahmen, die mehrere Jobs oder Geräte betreffen (nur Kommandos für einzelne Objekte),
   - die Arbeit mit privaten Prozedurbibliotheken,
   - die Möglichkeiten zur Steuerung der Ausgabe der Drucklisten insbesondere bei Havarien am Drucker (Papierriß oder -stau).

IV. Das OS/ES bietet für die optimale Ausnutzung der verschiedenen Ressourcen nur eine mehr oder weniger geringe Unterstützung. Eine effektive Ausnutzung ist in hohem Maße abhängig von der

Aufmerksamkeit und Qualität der Anlagenbedienung, von einer durchdachten und wirksamen Produktionstechnologie (Datenträger- und Dateiverteilung, Jobklassenverteilung u.a.) und vom Qualifikationsniveau der Nutzer. Schwierigkeiten bestehen bekanntlich im Ausgleich zwischen E/A- und zentraleinheitsintensiven Aufgaben, in der Ausnutzung des peripheren SYSDA-Direktzugriffsspeichers, vor allem aber hinsichtlich einer günstigen Verteilung der Nutzerdateien auf den Magnetdatenträgern (Speicherplatzausnutzung, Vermeiden des Wechselns von Datenträgern).

V. Das OS/ES bietet nur unzureichende bzw. kaum praktikable Unterstützung für den Schutz und die Sicherung der Dateien. Die konventionellen Möglichkeiten des Verfallsschutzes oder des Schutzes über Kennwort sind mit einem hohen Aufwand bei der Bedienung der EDVA verbunden und werden deshalb nur selten genutzt.

VI. In den letzten Jahren ist eine zunehmende Nutzung von Dialog- und Jobeingabesystemen zu beobachten. Auf Grund der räumlichen Trennung zwischen jobabsendenden Terminalnutzern und der empfangenden EDVA ist eine Aufgabenbeschreibung in Form einer Begleitkarte nicht möglich, sodaß die aus dem Fehlen dieser Begleitinformationen resultierenden Schwierigkeiten zur Zeit nur durch zusätzliche Restriktionen an die abgesandten Aufgaben abgefangen werden können.

Die genannten Probleme zur Automatisierung und effektiveren Organisation des Produktionsbetriebes an ESER-Anlagen unter dem OS/ES sind sehr komplexer Natur. Einige Aspekte der vorgenannten Schwierigkeiten sind durch Einzellösungen hier und da schon erkannt und bewältigt. Doch ist die Anwendung vieler einzelner Zusatzprogramme i.a. unrationell.

Davon ausgehend wurde in Zusammenarbeit zwischen Organisations- und Rechenzentren der Humboldt-Universität zu Berlin und der Martin-Luther-Universität Halle das System SPOOL als eine dem Betriebssystem OS/ES nebengeordnete Komponente zur Automatisierung, Produktivitätssteigerung und Rationalisierung des Produktionsprozesses an ESER-Rechnern entwickelt. Die Entwicklung dieses Systems erfolgte bisher in mehreren aufwärts kompatiblen Stufen SPOOL 1, SPOOL 2.0

und SPOOL 2.1.

Das Basissystem SPOOL 1 dient vorrangig der Rationalisierung der Anlagenbedienung, der Jobverwaltung und der effektiven Parallelarbeit der Zentraleinheit mit den externen Geräten. Es befindet sich seit 1980/81 in mehreren Hochschulrechenzentren im permanenten produktiven Einsatz. Die mit dem Einsatz des Systems verbundene Steigerung des Jobdurchsatzes ist abhängig von dem Anlagentyp, der Peripherie der Anlage und der Hauptspeichergröße und beziffert sich auf 5 bis 15 %.

Seit Januar 1982 wird die Version SPOOL 2.0 produktiv am Organisations- und Rechenzentrum der Humboldt-Universität mit dem Ziel eingesetzt, bei gleichbleibender Bedieneranzahl vom unterbrochenen Dreischichtbetrieb zum durchgängigen Dreischichtbetrieb überzugehen. Durch neue Komponenten zur Datenträgerverwaltung, Jobklassifizierung, Jobauswahl und zur Automatisierung der Konsolbedienung ist softwareseitig die Möglichkeit gegeben,

- die EDVA zeitweise (über mehrere Stunden) ohne Bedienereingriff zu betreiben,
- generell die Anlagenbedienung bezüglich der Magnetplattenauswahl zu optimieren und
- die Jobauswahl in Abhängigkeit von der Verfügbarkeit der benötigten Magnetdatenträger zu steuern.

In der Perspektive sind im Zusammenwirken mehrerer Hochschuleinrichtungen die Erweiterung des Systems SPOOL mit Datenträger- und Dateiverwaltungskomponenten (Bereitstellung, Auslagerung und Sicherung von Magnetplattendateien), die Unterstützung der lokalen Mehrrechnerkonfiguration durch SPOOL sowie der synchrone und asynchrone Anschluß von Stapelterminals als Endstellen eines sternartigen Rechnerverbundes vorgesehen. Der Einsatz des Systems SPOOL unter der Steuerprogrammkonfiguration SVS wird untersucht.

## 2. Leistungsumfang des Systems SPOOL

Das System SPOOL arbeitet als selbständige Aufgabe im Betriebssystem OS/ES (MVT, MFT). Dabei hat es sowohl bestimmte Arbeiten der Job-, Aufgaben- und der Datenverwaltung des OS/ES zu übernehmen als auch Funktionen der Automatisierung der Arbeitsvorbereitung sowie der

Arbeitsvorbereitung sowie der Rationalisierung und Automatisierung der Bedienung der EDVA wahrzunehmen.

Als eine Grundfunktion übernimmt SPOOL die Jobeingabe (Kartenleser, MB, MP) und die Jobausgabe (Drucker, Stanzer, MB). Dabei werden eine maximale Arbeitsgeschwindigkeit, eine zentrale rationelle Zwischenspeicherung der eingelesenen Jobs und der erzeugten Jobausgaben (keine Zerstückelung des SYSDA-Speichers), eine ständige Wiederanlauffähigkeit für Jobs, die sich in der Ausführungsphase befinden, und vor allem ein hoher Druckkomfort (Vor- und Rückwärtspositionieren, Unterbrechung, Wiederanlauf und Druckwiederholung) gewährleistet. Eine weitere Grundfunktion hinsichtlich der Jobverwaltung - die rationelle Gestaltung der Bedienung - wird durch die Kommandosprache von SPOOL erreicht. Durch entsprechende bequem anwendbare SPOOL-Kommandos werden die meisten OS/ES-Kommandos (insbesondere Kommandos, die die Jobverwaltung steuern oder Informationen über Jobketten usw. liefern) ersetzt und dabei zum Teil in ihrem Funktionsumfang erweitert. Die SPOOL-Kommandos aktivieren und steuern auch die zusätzlichen Funktionen des Systems SPOOL. Zur Unterstützung der Arbeit der Anlagenbedienung und ggf. der Arbeitsvorbereitung liefert SPOOL eine Jobinformationsliste (s. Abb. 1). In ihr sind die für Arbeitsvorbereiter und Bediener wichtigen Jobparameter verzeichnet. Dazu gehören:

- die intern verwendete Jobnummer, welche zur leichteren und schnelleren Bedienerkommunikation vom System SPOOL vergeben wird. Der Bediener kann in SPOOL-Kommandos diese Jobnummer (einzeln oder für ganze Gruppen) verwenden;
- Jobname,
- Jobklasse,
- Auftragsnummer,
- Programmierername,
- benötigter SYSDA-Arbeitsspeicher,
- Datenträgerarchivnummern,
- Namen bzw. Adressen direkt angesprochener Geräte,
- Namen der verfallsgeschützten Dateien, die überschrieben werden sollen;
- Wiederholbarkeit des Jobs sowie
- spezielle Anforderungen an die Bedienung.

Abb. 1

J O B  I N F O R M A T I O N  30.JUL.1982  11.52.26

| JNR | JOBNAME | PROGRAMMER | ACCTNGNR | CLS | TRKS | CPU | DUR | S | E | L | DISK | TAPE | UNIT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | HUBBIZ4K | KLAEMBT | 2-050-22 | C | STD | 015 | 40 | | | | HPM030 | HL1601 | |
| 2 | HUB030D | ARCHIVAR | 2-015-85 | C | STD | 005 | 25 | | | | HPM030 | IL1861* | |
| | | | | | | | | | | | | HL1862* | |
| 3 | HUBBIZ5K | KLAEMBT | 2-050-20 | C | STD | 020 | 40 | | | | HPM030 | HL1372 | |
| 4 | HUBFTH1B | ARCHIVAR | 2-015-85 | A | STD | 005 | 10 | | | | | HL2238* | |
| 5 | HUBFTH1B | ARCHIVAR | 2-015-85 | A | STD | 005 | 10 | | | | | HL2238* | |
| 6 | HUBSYM67 | STOECKEL | 2-014-10 | C | STD | 015 | 30 | | | | HPM030 | | |
| 8 | HUBSYM99 | STOECKEL | 2-014-10 | C | STD | 015 | 30 | | | | HPM030 | | |
| 9 | HUBSYM60 | STOECKEL | 2-014-10 | C | STD | 015 | 30 | | | | | | |
| 10 | HUBSYM60 | STOECKEL | 2-014-10 | C | STD | 015 | 30 | | | | | | |
| 11 | HUBBPH01 | GUENTHER | 2-039-1 | H | STD | 020 | 40 | | | | HPM031 | HL2031* | 6030 |

VOLUME TABLE INFORMATION 30.JUL.1982 11.52.26

| VOLUME SERIAL | WITH OPERATOR DUR | JOBS | NO OPERATOR DUR | JOBS | SUMMARY DUR | JOBS |
|---|---|---|---|---|---|---|
| HPM041 | 0 | 0 | 41 | 1 | 41 | 1 |
| HPM050 | 0 | 0 | 0 | 0 | 0 | 0 |
| HPM033 | 25 | 1 | 41 | 1 | 66 | 2 |
| HPM032 | 20 | 1 | 0 | 0 | 20 | 1 |
| 036 | 0 | 0 | 0 | 0 | 0 | 0 |
| HPM048 | 0 | 0 | 0 | 0 | 0 | 0 |
| HPM049 | 40 | 1 | 0 | 0 | 40 | 1 |
| HPM036 | 80 | 1 | 0 | 0 | 80 | 1 |
| HPM031 | 15 | 0 | 0 | 0 | 15 | 0 |

END OF VOLUME TABLE INFORMATION

Um dem Bediener die Konfigurierung der Magnetplatten zu erleichtern, enthält die Jobinformationsliste zusätzlich zu jeder Magnetplatte, die von Jobs benötigt wird, Angaben über die Anzahl und die geschätzte Laufzeit aller sich auf sie beziehenden Jobs. Alle diese Angaben werden der JOB-Anweisung des OS/ES und speziellen, durch die Nutzer bereitzustellenden SPOOL-Steueranweisungen (s. 3.3.) entnommen. Eine weitere wesentliche Funktion, die ebenfalls schon in der Version SPOOL 1 realisiert ist, ist die Prioritätensteuerung. SPOOL steuert die Auswahlpriorität zur Verarbeitung (Jobpriorität), die Auswahlpriorität während der Verarbeitung (Verarbeitungspriorität) und die Auswahlpriorität zur Jobausgabe (Ausgabepriorität). Bei der Verwendung von Standardwerten bei der Generierung des Systems SPOOL wird die Jobpriorität in Abhängigkeit von den durch den Nutzer in einer SPOOL-Steueranweisung kodierten geschätzten Joblaufzeit und Druckzeilenanzahl so gebildet, daß der Expreßbetrieb bevorzugt wird. SPOOL veranlaßt, daß niedrige Werte für Laufzeit und Druckzeilen zu einer hohen Priorität führen und umgekehrt. Es ist möglich, daß der Bediener während der Abarbeitung des Jobs bei Überschreiten der Schätzwerte zyklisch informiert wird und daß generierungsabhängig im gegebenen Fall das Beenden des Jobs (automatisches CANCEL) durch das System bei Überschreiten der Schätzwerte veranlaßt wird. Die Bevorzugung des Expreßbetriebes wird standardmäßig auch bei der Bildung der Ausgabepriorität durchgeführt, wobei als bestimmender Parameter die tatsächliche Druckzeilenzahl des Jobs verwendet wird. Während der OS/ES-Verarbeitung der Jobs überwacht das System SPOOL deren E/A- und Zentraleinheitsbelastung und modifiziert dynamisch in festgelegten Zeitabständen die Verarbeitungspriorität der Aufgaben derart, daß Jobs mit einer hohen E/A-Rate und niedriger Zentraleinheitsbelastung bevorzugt werden.

Ein weiterer Komplex von Funktionen, der ab der Version SPOOL 2.0 eingeschlossen ist, befaßt sich mit der Automatisierung der Anlagenbedienung, insbesondere mit den Möglichkeiten, die EDVA zeitweise ohne Bedienereingriff zu betreiben.

Durch das System werden drei verschiedene, durch Kommando zu aktivierende Betriebsweisen realisiert, deren wesentliche Unterschiede vor allem in einer Jobauswahl liegen:

B1 Abarbeitung von Jobs, die Bedienereingriffe erfordern;

B2 Abarbeitung von Jobs, die keine Bedienereingriffe erfordern;
B3 Abarbeitung aller Jobs - unabhängig von ihren Bedienanforderungen.

In allen drei Betriebsweisen gelangen standardmäßig nur solche Jobs zur Ausführung, deren benötigten Magnetplatten dem System durch ein spezielles Bedienkommando als logisch verfügbar gekennzeichnet wurden.

Das System SPOOL 2.1 befaßt sich zusätzlich mit der Unterstützung solcher Funktionen, die Arbeitsgänge der Arbeitsvorbereiter automatisiert. Durch die Verwaltung eines Katalogsystems ist SPOOL in der Lage, die Berechtigung zur Inanspruchnahme folgender Ressourcen durch die Jobs zu überprüfen: Rechenzeit, Druckerpapierverbrauch, Magnetdatenträger und Dateien. Die beabsichtigte unberechtigte Nutzung von Ressourcen, z.B. bei überschrittenem Rechenzeit- oder Papierkontingent oder bei Zugriff zu Dateien, für die der Nutzer (Auftragsnummer) nicht autorisiert ist, führt zum Entfernen des entsprechenden Jobs (automatisches CANCEL) aus dem System, bevor seine eigentliche Abarbeitung beginnt. Durch die Überwachung des Dateizugriffs wird die Anwendung der OS/ES-Schutzmechanismen überflüssig und es entfallen somit auch diesbezügliche Bedieneraktivitäten.

Durch eine Angleichung der entsprechenden TSO-Komponenten /2/ ist es möglich, die Systeme TSO und SPOOL parallel laufen zu lassen. Es wurde abgesichert, daß delegierte Jobs sich in das technologische Schema einpassen können und somit wie Jobs behandelt werden, die über die Auftragsannahme an die Anlage gelangen.

Das SPOOL-System besitzt als Front-End-Prozessor auch die Möglichkeit, von intelligenten Stapelterminals, Jobfernverarbeitung zu betreiben. Zur Zeit liegen nur für solche Terminals Anschlußmöglichkeiten vor, die über Synchronleitungen angeschlossen werden können /3/.

Weitere Entwicklungen im Rahmen der SPOOL-Bearbeitung sind:

- die Realisierung einer lokalen Mehrrechnerkonfiguration, die die lokale Kopplung von zwei ESER-Rechnern über einen Kanal-Kanal-Adapter unter der Bedingung einer Bedienbarkeit von einer Anlage aus bieten soll,
  und

- die Automatisierung der Dateibereitstellung, die der rationellen Verwaltung großer WPS unter Zuhilfenahme von Hintergrundspeichern (MB-Archiv) dienen soll und die sich als SPOOL 3 in der Realisierungsphase befindet /4/.

Zusammengefaßt ergeben sich bei Einsatz der bereits fertiggestellten Teile des Systems SPOOL folgende Effekte:

- Erhöhung des Jobdurchsatzes;
- starke Verringerung des Arbeitsaufwandes in der Arbeitsvorbereitung;
- erhöhte Sicherheit der Ressourcennutzung;
- effektivere Arbeit des Bedienpersonals;
- Durchführung eines zeitweise bedienerlosen Betriebes und damit
- Verringerung des Aufwandes an Bedienpersonal bzw. Verlängerung der produktiven Einschaltzeit der ESER-Anlage bei gleichbleibender Anzahl der Bediener.

## 3. Aufbau und Arbeitsweise des Systems SPOOL

### 3.1. Bestandteile und Struktur des Systems SPOOL

Die Funktionen des Systems SPOOL werden durch eine Reihe von Prozessoren realisiert, die durch einen zentralen Dispatcher über eine interne WALT-POST-Logik mit Zentraleinheitszeit versorgt werden. Im wesentlichen handelt es sich dabei um die Prozessoren

- Eingabeprozessor(en) (RDR) für die Realisierung der Jobeingabe;
- Prüfprozessoren (CHK) für die Ressourcenprüfungen;
- Initiatorprozessoren (XEQ) für die Jobauswahl und die Absicherung weiterer Maßnahmen während der Jobabarbeitung (z.B. Zugriff zur sog. SPOOL-Queue);
- Aufgabenüberwachungsprozessor (XMON) für die dynamische Prioritätensteuerung;
- Aktualisierungsprozessor (ACTU) für die Aktualisierung des SPOOL-Kataloges;
- Prüfpunktprozessor (CKPT) für die Sicherung von Wiederanlaufinformationen von SPOOL-Aktivitäten;
- Ausgabeprozessor(en) (PRPU) für die Ausgabe der Jobergebnisse auf Drucker, Stanzer und (Drucklisten-)Magnetband;

- Kommandoprozessor (COMM) für die Umsetzung der Kommandosprache;
- Konsolprozessor (CON) für die Unterstützung der Konsolbedienung;

Das System SPOOL besitzt eine spezielle Überlagerungsorganisation, die es erlaubt, die Funktionen mit einem relativ geringen Hauptspeicherbedarf zu realisieren (s. 4.5.).

Zur Absicherung seiner Funktionen startet SPOOL zwei (MFT) bzw. drei (MVT) Unteraufgaben in seinem Programmbereich. Darüber hinaus wird jeweils unmittelbar nach der Jobauswahl ein besonders modifiziertes Systemeingabeprogramm (OSRDR) initiiert. Im MFT arbeitet der OSRDR als S-transiente Aufgabe, im MVT wird er durch eine gesonderte Unteraufgabe und den interpretierenden Teil des ASE-Readers realisiert.

Die zentrale Datei des Systems SPOOL ist die sogenannte SPOOL-Queue. Diese stark frequentierte Datei dient zur Aufbewahrung der Jobinformationen (Jobstrom, -ausgaben, Steuerblöcke) und wird auf einem oder auf mehreren Magnetplatten angelegt.

## 3.2. Jobdurchlauf durch das System SPOOL

SPOOL kann Jobströme von Kartenlesern, sowie von Magnetplatten- und Magnetbanddateien einlesen. Außerdem ist es Nutzerjobs oder interaktiven Systemen über die Komponente "Interner Reader" leicht möglich, Jobs in die SPOOL-Queue zu delegieren. Die in die SPOOL-Queue gelesenen Jobs verbleiben dort in ihrer ursprünglichen Form bis zum Ende der Ausführphase, sodaß ggf. Jobwiederanläufe (Restart oder Warmstart) möglich sind.

Während der Jobeingabe werden die SPOOL-Steueranweisungen (s. 3.3.) analysiert, die für die Erzeugung der Jobinformationsliste notwendigen Angaben in einer Hilfsdatei abgespeichert und die Jobpriorität festgelegt. Nach der Jobeingabe wird mittels der SPOOL-Katalogdateien die Verfügbarkeit der Ressourcen Rechenzeit, Papier und Datenträger geprüft. Ein negatives Ergebnis führt zum Entfernen des Jobs aus dem System.

Die von den akzeptierten Jobs benötigten Magnetplattenarchivnummern werden in eine interne Tabelle (VOLTAB) eingetragen, wodurch eine datenträgerabhängige Jobauswahl und entsprechende Informationen in

der Bedienerinformationsliste und in Displaykommandos ermöglicht werden. Außerdem wird vom System SPOOL untersucht, ob der Job ohne Bedieneraktivitäten verarbeitet werden kann oder nicht.

Die Auswahl der Jobs zur Verarbeitung wird von SPOOL-Initiator-Prozessoren vorgenommen, die jeweils genau einem OS/ES-Initiator zugeordnet sind (Region bzw. Position). Die Auswahl erfolgt dabei in üblicher Weise über Jobklasse und Priorität, wobei die Betriebsweise und das Ergebnis der internen Jobklassifizierung mit Einfluß nehmen. Der Initiator übergibt den ausgewählten Job dem OS/ES, indem er einem Systemeingabeprogramm, dem sog. OSRDR, übergeben wird. Dieser überprüft die Syntax, modifiziert den Job für interne Zwecke und stellt ihn in der Systemkettendatei des OS/ES zur Abarbeitung bereit.

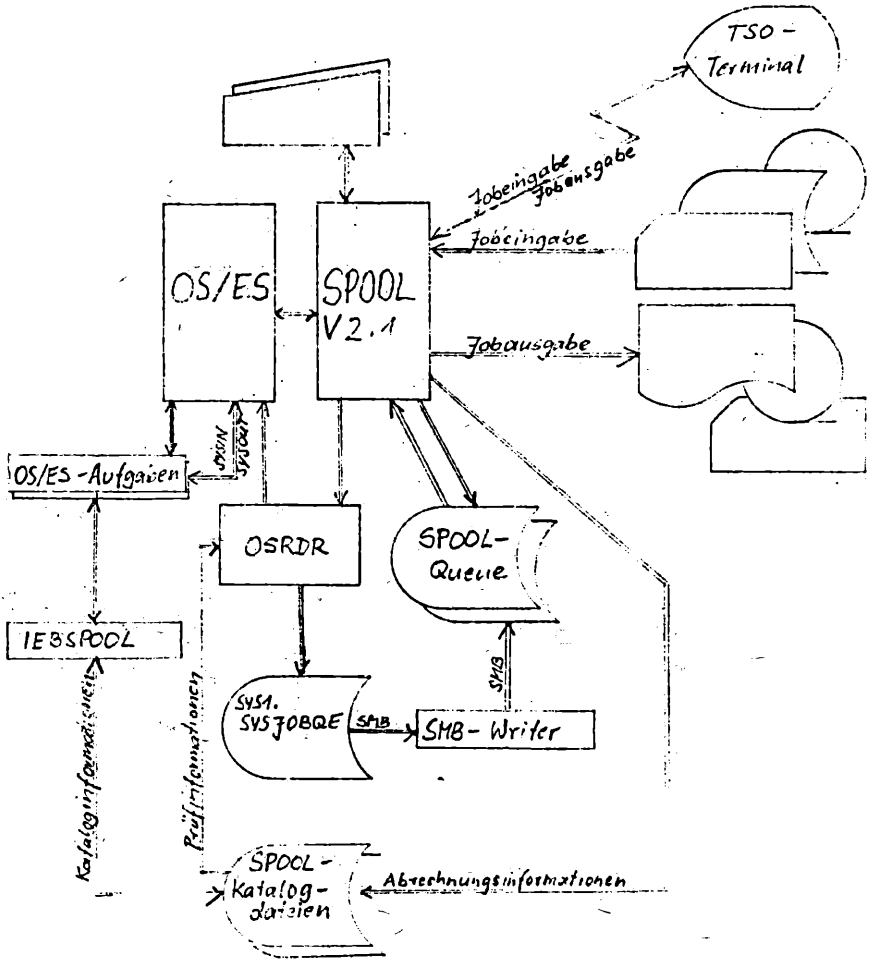
Bevor die normale Abarbeitung des Jobs im OS/ES beginnt, wird mittels der SPOOL-Katalogdateien geprüft, ob die Verwendung der Magnetplattendateien in zulässiger Weise erfolgt. Ein negatives Prüfergebnis führt zum Entfernen (automatisches CANCEL) des Jobs aus dem System; andernfalls erfolgt die Freigabe (automatisches RELEASE) des Jobs.

Während der Jobabarbeitung organisiert das System SPOOL die Eingabe der Eingabestromdateien und die Ausgabe der SYSOUT-Dateien gewisser (generierungsabhängig) Ausgabeklassen mit Hilfe der vorgenommenen, nur intern wirksamen Jobmodifikationen in solcher Weise, daß die Eingaben von der SPOOL-Queue und die Ausgaben in die SPOOL-Queue erfolgen.

Nach Beendigung der Arbeit des Jobs werden dessen Jobnachrichten (SMB-Informationen) aus dem OS/ES in die SPOOL-Queue übernommen. Damit stehen alle Ausgabeinformationen des Jobs in der SPOOL-Queue zur Ausgabe bereit.

Durch Ausgabeprozessoren werden sie auf Drucker, Stanzer oder Magnetband ausgegeben. Abbildung 2.

Abb. 2 Joblaufschema (stark vereinfacht) mit SPOOL 2.1



## 3.3. Kommunikation SPOOL-Nutzer

Das System SPOOL und die Nutzer (Problemanalytiker, Programmierer) kommunizieren durch

- die vom Nutzer in den Jobs zu kodierenden SPOOL-Steueranweisungen, die neben Steuerparametern eine Auftragsbeschreibung (ähnlich der Jobbegleitanweisung) enthalten;
- die Jobausgaben, insbesondere durch die von SPOOL bereitgestellte JOB-LOG, die Konsolnachrichten enthält, die den jeweiligen Job betreffen.

Die SPOOL-Steueranweisungen beginnen mit der Zeichenfolge "/*", gefolgt von dem jeweiligen Kodewort. Auf Grund dessen wirken sie als Kommentaranweisungen, wenn SPOOL nicht aktiv ist. Im einzelnen gibt es folgende Anweisungen:

- /*JOBPARM-Anweisung unter anderem mit Schätzwerten für Joblaufzeit, Druckzeilenanzahl und benötigten SYSDA-Speicherplatz für Arbeitsdateien sowie mit Angaben zur Jobwiederholbarkeit, speziellen Bedieneraktivitäten (WTOR) und Kopieanzahl der Jobliste;
- /*SETUP-Anweisung mit Angabe der direkt angesprochenen Geräte, der Magnetdatenträger und der zu überschreibenden verfallsgeschützten Dateien einschließlich zusätzlicher Informationen wie Schreibringmontage bei Magnetbändern oder EOR-Zeichen bei Lochstreifengeräten;
- /*PRIORITY-Anweisung mit Festlegung der Jobpriorität;
- /*MESSAGE-Anweisung mit Informationen, die zum Zeitpunkt der Jobeingabe an den Anlagenbediener übermittelt werden.

Die von SPOOL ausgegebenen Jobausgabelisten weisen vor allem folgende Besonderheiten auf:

- Auf der Separatorseite des Jobs erscheinen zusätzlich die schon erwähnte JOBLOG sowie statistische Informationen (Laufzeit, Druckzeilenanzahl, Drucklistentypen u.a.).
- Die Jobnachrichten (JCL-Anweisungen usw.) erscheinen konzentriert am Anfang der Jobliste unmittelbar nach der JOBLOG.
- Nach den Jobnachrichten folgen die Ausgaben der Jobsystemausgabeklasse, wobei Dateiende und Jobende durch entsprechende Nachrichtenzeilen gekennzeichnet sind.

- Unterbrochene Joblisten bzw. Listen verschiedener Ausgabeklassen werden mit einer neuen Separatorseite begonnen.

## 3.4. Kommunikation SPOOL - Bediener

Die Kommandosprache des Systems SPOOL zeichnet sich durch Einfachheit, Kürze und durch die Verwendung von Listenkommandos, also das Verwenden eines Kommandos und die anschließende Ausführung desselben für eine Gruppe von Jobs (mehrere Jobnummern), Geräten oder Datenträgern, aus.

Kommandos und Nachrichten beginnen mit einem Währungszeichen ($). Der Kommandokode ist auf weitgehende Ähnlichkeit zu vergleichbaren OS/ES-Kommandos abgestimmt.

Die SPOOL-Kommandos lassen sich in fünf Hauptgruppen einteilen:

A. Systemkommandos
B. Displaykommandos
C. Jobkommandos
D. Gerätekommandos
E. Datenträgerkommandos.

Viele dieser Kommandos können im Sinne einer Listenform auf mehrere Objekte gleichzeitig angewendet werden. Als Beispiel seien folgende Kommandos angeführt:

S A J11-20,26-27 - Freigabe der Jobs mit der internen Jobnummer 11, 12, ... bis 20, sowie 26 und 27.

S S PRT1,RDR2,TPE2 - Starten der vorübergehend gestoppten Geräte Drucker 1 (PRT1), Kartenleser 2 (RDR2) und Magnetbanddruckausgabe 2 (TPE2).

S M 111111,134,222222,137 - logisches Verfügbarmachen (SPOOL-MOUNT) der Platten mit der Archivnummer 111111 auf Gerät 134 und 222222 auf Gerät 137 (intern wird ein OS/ES-MOUNT wirksam).

# 4. Stellung des Systems SPOOL zum Betriebssystem OS/ES

## 4.1. Forderungen an das Betriebssystem OS/ES

Das Betriebssystem OS/ES (MFT/MVT) muß für die Arbeit von SPOOL durch eine vollständige oder eine E/A-Generierung vorbereitet werden. Dabei müssen u.a.

- eine gewisse Anzahl Pseudo-UR-Geräte für die Realisierung der SPOOL-Queue vorgesehen werden;
- die Gerätegruppennamen A und INTRDR im System verfügbar sein;
- der Typ 1 SPOOL-SVC eingeschlossen werden.

Das so modifizierte Betriebssystem ist auch ohne SPOOL uneingeschränkt arbeitsfähig.

## 4.2. Generierung des Systems SPOOL

Das System SPOOL kann durch eine Vielzahl von Parametern variabel an die Gegebenheiten und Wünsche des Anwenderrechenzentrums angepaßt werden. Die Generierung wird durch Systemprogrammierer mittels Auswahl der Generierungsparameter, z.B. Auswahl der Anzahl der SPOOL-Initiatoren, der zu unterstützenden SPOOL-Geräte, der Puffer, Jobklassen usw. oder aber Auswahl der Standardparameter für die JOBPARM-Anweisung oder Festlegen der Konsequenzen beim Übertreten der zulässigen Nutzungsbedingungen (Dateinutzung, Laufzeitüberschreitung usw.) vorbereitet. Kriterien sind der gewünschte Funktionsumfang, der Umfang der zu unterstützenden Ressourcen und der Speicherbedarf.

## 4.3. Aufbau und Wartung der SPOOL-Katalogdateien

Für die Kontrollfunktionen von SPOOL ist ein Komplex von Katalogdateien vorhanden, der aus

- dem Zentralen Katalog (KAT),
- dem Dateiverzeichnis (DSN),
- der Auftragsdatei (ACC),
- der Datenträgerdatei (VOL) und
- Hilfsdateien

besteht, die untereinander verkettet sind.

Für den Aufbau und die Pflege der Katalogdateien steht das Dienstprogramm IEBSPOOL zur Verfügung /3/.

## 4.4. Systemstart und -beendigung

Wie schon erwähnt arbeitet SPOOL als Aufgabe des Betriebssystems OS/ES. Während der Initiierungsphase werden zunächst von SPOOL im Betriebssystem temporäre Veränderungen vorgenommen, die die Arbeitsfähigkeit von SPOOL sichern (z.B. Modifikation der SVC-Tabelle). Außerdem werden die Unteraufgaben und die OS/ES-Initiatoren gestartet und die SPOOL-Queue wird auf ihre Verwendbarkeit geprüft. Der Bediener hat die Wahl zwischen einem SPOOL-Warmstart und dem SPOOL-Kalt-/Formatstart. Beim Warmstart werden Jobs, die noch im System sind, zum Wiederanlauf geführt, den Ausgabeprozessoren zugeführt oder im normalen Modus der prioritätsgesteuerten Abarbeitung unterzogen.

Wird dem Bediener der Ruhestand des Systems angezeigt, d.h. es ist kein Job mehr aktiv und kein Gerät ist in Arbeit, so kann er mit dem Kommando S PSPOOL die Beendigung der Arbeit des Systems SPOOL einleiten. Hierbei werden dann alle temporären Änderungen im OS/ES wieder rückgängig gemacht, sodaß das OS/ES wieder normal arbeitsfähig ist.

## 4.5. Speicherbedarf

Der Hauptspeicherbedarf des Systems SPOOL ist von der Auswahl der Generierungsparameter und damit im wesentlichen von der Hauptspeichergröße und von der Gerätekonfiguration der Anlage abhängig. Für mittelgroße (512 K Bytes) und große Anlagen kann man unter der Steuerprogrammkonfiguration MVT mit 60 bis 80 K Bytes rechnen. Für die Steuerprogrammkonfiguration MFT vermindert sich der Bedarf um etwa 8 bis 10 K Bytes. Es sei darauf hingewiesen, daß zusätzliche Systemeingabeprogramme und Systemausgabeprogramme nicht benötigt werden, sodaß sich bei Abzug von 40 K Bytes (residenter Teil des ASB-Readers und ein Systemausgabeprogramm) eine tatsächliche Speichermehrbelastung von 20 bis 40 K Bytes bei Einsatz des Systems SPOOL ergibt.

Der Mehrbedarf beim Steuerprogramm des OS/ES ist in Abhängigkeit von der Menge der notwendigen Pseudogeräte auf ein bis zwei K Bytes zu

veranschlagen.

Literaturhinweise

/1/ Kroß, Günther:
SPOOL 2.0 - System zur Unterstützung eines zeitweise bedienungsunabhängigen Betriebes an ESER-Anlagen

/2/ Weickmann, Christoph:
Gemeinsame Nutzung von SPOOL und TSO auf einer EDVA

/3/ Meißler, H.G., Fenske, B.:
Möglichkeiten der Jobferneingabe in das AV-System SPOOL

/4/ Piens, Conrad:
Effektivitätsuntersuchungen für das System SPOOL 3

Wenzlaff, Bodo

# Gesellschaftliche und soziale Auswirkungen der EDV-Anwendung

Die Analyse sozialer und gesellschaftlicher Wirkungen, die mit der Anwendung der EDV-Technologien verbunden sind, hat insbesondere in den letzten Jahren wachsende Beachtung gefunden. Nach anfänglicher Euphorie und bedingungsloser Zukunftsgläubigkeit ist in den westlichen Ländern der kritische und pessimistische Charakter der Technologiebewertung immer mehr in den Vordergrund gerückt. Viele sehen in der rasanten Technikentwicklung der letzten Jahrzehnte die entscheidende Ursache für die immer sichtbarer werdenden gesellschaftlichen und sozialen Deformationen unserer Tage.
So meint beispielsweise W. Steinmüller: "Es droht heute nicht nur eine atomare und chemische Umweltverschmutzung, sondern ebenso die automationsunterstützte Denaturierung der psychosozialen Umwelt und Inwelt - nur mit dem Unterschied, daß in diesem Bereich die Entwicklung wesentlich schneller verlaufen wird als bei der Industrialisierung der körperlichen Arbeit. Deshalb handelt es sich um eine Gegenwartsaufgabe, die nicht auf ungewisse Zukünfte vertagt werden darf."
So etwa könnte man die kritische Grundrichtung der modernen bürgerlichen Wirkungsforschung in Kurzform zusammenfassen.

Während in den kapitalistischen Ländern die Wirkungsforschung bereits eine relative Breite erreicht hat, unterschiedliche theoretische Ausgangspunkte erkennbar sind und zu entsprechenden Schulen und Richtungen geführt haben, stehen wir im Sozialismus noch ganz am Anfang einer wissenschaftlich begründeten Analyse der mittel- und langfristigen Auswirkungen bei der Anwendung moderner Technologien, wie sie insbesondere durch die Computertechnik und Mikroelektronik bereitgestellt werden. Zum Teil fehlt es noch überhaupt an dem notwendigen Problembewußtsein, weil man offensichtlich glaubt, daß sich unter sozialistischen Produktionsbedingungen automatisch jeder wissenschaftlich-technische Fortschritt zum Besten der Menschen auswirken müsse.

Sicherlich gibt es für uns keinen Anlaß zu pessimistischer Schwarzmalerei, wenn wir uns mit hohem Engagement der schnellen Entwicklung und Anwendung moderner Informationstechnologien zuwenden. Dennoch sind die hierdurch bewirkten gesellschaftlichen und sozialen Wirkungen keineswegs problemlos.
Bevor wir uns den einzelnen Wirkungen und ihrer Einschätzung zuwenden, wobei insbesondere Prof. Fuchs-Kittowski zu den verschiedenen Strategien der Anwendung von Informationstechnologien im Hinblick auf die Ergebnisse moderner Wirkungsforschung noch sprechen wird, wollen wir im folgenden versuchen, uns über einige Grundfragen der Zielstellung und möglichen Ergebnisse einer marxistisch-leninistisch orientierten Wirkungsforschung verständigen.
Das ist deshalb so wichtig, weil durch die bürgerliche Wirkungsforschung suggeriert wird, daß soziale und gesellschaftliche Auswirkungen der EDV-Anwendung prinzipiell negativ sind.
Die Identifizierung von Wirkungen mit negativen Auswirkungen ist schon deshalb abzulehnen, weil eine solche Grundhaltung zwangsläufig zu einer konservativ-romantischen Kritik gegenwärtiger Entwicklungsprozesse führen muß. Die Wurzeln hierfür liegen unseres Erachtens in der unlösbaren Krisenhaftigkeit des kapitalistischen Gesellschaftssystems, wodurch jede Entwicklung als Verschärfung bestehender Widersprüche und folglich als sich beschleunigender Niedergang auch im theoretischen Denken reflektiert wird.
Aus genau diesen Gründen können wir daher nicht unbesehen die sogenannten Erfahrungen der westlichen Länder bei der Einführung von EDV-Systemen übernehmen. Gerade in der Wirkungsforschung und ihrer Einordnung, die mehr und mehr zu einem integrierenden Bestandteil der Informatik als Wissenschaftsdisziplin selbst geworden ist, zeigt sich der gesellschaftswissenschaftliche Aspekt dieser neuen Wissenschaft.

Jede Entwicklung - das zeigt sich bereits in der Biologie - ist irreversibel, dh., daß eine Rückkehr zu früheren Ausgangspunkten der Entwicklung nicht mehr möglich ist. Es gehört gewissermaßen zum Wesen der Entwicklung, daß sie in die Zukunft gerichtet ist.

Um so wichtiger ist es, Entscheidungen, die nachhaltige Auswirkungen auf Entwicklungsprozesse haben, besonders gründlich zu überdenken.

Ein weiterer Aspekt dieses Evolutionsverständnisses ist aber auch, die Ambivalenz jeglicher Art von Entwicklungsprozessen zu begreifen und in die Gestaltungsstrategien einzubeziehen. Ambivalenz heißt, daß Fortschritt nicht nur Verlängerung alles bisher Guten und Positiven und Verringerung alles bisher Schlechten und Negativen ist und sein kann, sondern daß die objektiven Entwicklungszusammenhänge auch manches Liebgewordene und nicht unbedingt Negative unter den Hammer bringen.
Fortschritt ist ein objektiver Prozeß, der sich von den Wünschen der Menschen unterscheidet. Nur dort, wo das Gewünschte mit dem objektiv Realisierbaren in Übereinstimmung zu bringen ist, werden zusätzliche Impulse eines subjektiven Engagements für die Durchsetzung des Neuen ausgelöst, während umgekehrt Diskrepanzen zwischen dem wünschenswert Erscheinenden und dem objektiv Notwendigen diese subjektiven Antriebe lähmt. In der marxistisch-leninistischen Ideologie widerspiegelt sich dieser Sachverhalt in dem Bemühen, die objektiven Erfordernisse gesellschaftlicher Entwicklung so überzeugend darzulegen, daß sie nicht nur verstanden, sondern möglichst auch in persönliche Haltungen und Lebensziele umgesetzt werden.

Die Beurteilung von Auswirkungen der EDV-Anwendung hängt aus den o.g. Gründen ganz wesentlich von den gesellschaftstheoretischen Grundpositionen ab, die der Wissenschaftler explizit oder implizit in die Wirkungsforschung der EDV einbringt.

Fassen wir zunächst die beiden Ausgangspunkte unserer weiteren Überlegungen noch einmal zusammen:

1. Jede Entwicklung ist irreversibel.
2. Jede Entwicklung ist ambivalent.

Wenn man von Auswirkungen des Einsatzes der EDV-Technologien spricht, muß man natürlich präzisieren, welche Teilprozesse primär am stärksten betroffen werden und welcher Zusammenhang zwischen direkten und indirekten Wirkungen besteht. Auch unter diesem Blickwinkel ergibt sich die Notwendigkeit, ein wissenschaft-

lich begründetes Bild über das Zusammenspiel von Teilprozessen in der Gesellschaft der Einordnung und Bewertung der Informationstechnologien zugrunde zu legen.
Anderenfalls kommt man über rein phänomenologische Beschreibungen nicht hinaus. Das ist meines Erachtens auch einer der Hauptgründe, warum die bürgerliche Technologiekritik so überaus vielfältig und buntscheckig ist. Neben naiven Gesellschaftsidyllen, die beschworen werden, findet man auch recht abenteuerliche Behauptungen über angebliche Prinzipien und Gesetzmäßigkeiten, nach denen sich menschliches Leben in der Gesellschaft entfaltet. Je nach den Ausgangspunkten dieser meist implizit und intuitiv vorgetragenen Überzeugungen von Wesen der Gesellschaft unterscheiden wir die mehr moralisierenden und appelierenden voluntaristischen Richtungen von den kausal analytischen und funktionalistisch beschreibenden.
Gemeinsam ist all diesen Richtungen, daß aus den Tiefen des sog. menschlichen Wesens, das angeblich durch die technische Entwicklung der letzten Jahrhunderte deformiert wurde, die prinzipielle Wirkungsweise der Informationstechnologien in Gegenwart und Zukunft erklärt wird. Es soll jedoch auch nicht unerwähnt bleiben, daß hierbei auch immer deutlicher die wirklichen Ursachen beim Namen genannt werden, jedoch meist erst von den Verbrämungen befreit werden müssen, die ihnen schon deshalb anhaften, weil sie bürgerlich-salonfähig sein wollen und sein müssen. So erscheint die Technologieentwicklung und insbesondere der Einsatz moderner Informationstechnologien primär als ein generelles Menschheitsproblem und - erst davon abgeleitet - als ein konkretes gesellschaftlich bedingtes Klassenproblem.

Die marxistisch-leninistische Analyse dieser Sachverhalte geht genau umgekehrt davon aus, daß alle relevanten Faktoren und Probleme der Gesellschaftsentwicklung primär in ihrem Zusammenhang mit den Interessen herrschender Klassen zu untersuchen sind und erst nach Klärung und Herausarbeitung dieser Zusammenhänge die allgemein menschlichen Implikationen moderner Technik und Technologie analysiert und eingeordnet werden können. Bei der Einschätzung, worauf die Technologien wirken, welche Veränderungen sie bewirken, sind folglich die Prozesse in den Mittelpunkt zu stellen, die die Gesellschaftsstruktur maßgeblich de-

terminieren. Das sind, aus marxistisch-leninistischer Sicht, in erster Linie der Arbeitsprozeß, die Organisationsstruktur, der Lebensprozeß und die Entfaltung und Entwicklung der Wesenskräfte des Menschen, wobei unter Mensch nicht nur das Individuum, die Person, zu verstehen ist, wie die bürgerliche Ideologie Glauben machen will, sondern das Ensemble seiner wesentlichen Beziehungen und Verhältnisse.

Sowohl historisch als auch vom Umfang der Analysen her gesehen, stehen die Wirkungen der Informationstechnologie auf den Arbeitsprozeß im Mittelpunkt der Beurteilungen und Bewertungen. Durch die alleinige Konzentration auf den gesellschaftlichen Arbeitsprozeß entstehen gleichzeitig jedoch Gefahren für Fehlinterpretationen, weil man versucht ist, auch die nicht hierher gehörenden Wirkungen in diesem Aspekt einzuordnen und allein hieraus zu erklären. Der Arbeitsprozeß ist schon deshalb Ausgangspunkt, weil er das direkte Einsatzgebiet der Informationstechnologien ist, um Steigerungen der Arbeitsproduktivität und eine hohe Gesamteffektivität der Produktion zu erzielen. Die erste Schwierigkeit, die sich auch terminologisch zeigt, ist die präzise Unterscheidung der gesellschaftlichen und sozialen Aspekte der untersuchten Wirkungen. Meist werden die gesellschaftlichen und sozialen Wirkungen als identisch betrachtet. Aus der Sicht der marxistisch-leninistischen Gesellschaftstheorie empfiehlt es sich jedoch, die Differenzierungsmöglichkeiten, die diese Termini bieten, auch in sinnvoller Weise zu nutzen. Da die Diskussionen einer solchen zweckmäßigen Unterscheidung noch in vollem Gange sind, soll an dieser Stelle nur gezeigt werden, in welcher Weise zwischen gesellschaftlich und sozial unterschieden werden könnte, um die verschiedenen Wirkungsrichtungen zu kennzeichnen:

Unter sozialen Wirkungen soll im folgenden immer ein solcher Wirkungsaspekt verstanden werden, der sich primär auf die beteiligten Individuen bezieht, wobei allerdings unter Individuum nicht der einzelne Mensch schlechthin, sondern die durch die sozialen Bedingungen seiner Existenz determinierte Persönlichkeitsstruktur gemeint ist. Sozial am und im Arbeitsprozeß sind also die auf den Menschen bezogenen Arbeitsbedingungen, die von

den elementaren Voraussetzungen, überhaupt Arbeit zu haben, bis zu den komplexeren Bedingungen reichen, welche Möglichkeiten der Arbeitsprozeß bietet, um menschliche Persönlichkeit zu entwickeln und zu entfalten. Hiermit ist der große Themenkreis der viel diskutierten sogenannten Entfremdung oder der Möglichkeiten des Menschseins im Arbeitsprozeß umrissen.

Der Einsatz der Informationstechnologien im Arbeitsprozeß verändert in gravierender Weise sowohl den Inhalt als auch die Form menschlicher Tätigkeit. Durch die sich in großen Dimensionen vollziehenden Rationalisierungsprozesse werden ganze Bereiche bisher menschlicher Tätigkeit durch die neue Technik selbst realisiert. Menschliche Tätigkeit wird also ersetzt und dadurch das gesamtgesellschaftliche Problem einer Neusetzung und Umprofilierung bisheriger menschlicher Tätigkeiten gestellt. Im Kapitalismus erfolgt dieser Prozeß als Durchsetzung von Profitinteressen und daher unmittelbarer und brutaler als unter sozialistischen Produktionsverhältnissen. Er zeigt sich vor allem darin, daß selbst bei leicht steigender Gesamtproduktivität die zunehmende Massenarbeitslosigkeit zu einem scheinbar unabwendbaren Begleitumstand der Expansion der EDV und Mikroelektronik geworden ist. Dieser ohne jegliche theoretische Reflexionen greifbare Sachzusammenhang verführt sehr leicht zu der Fehlinterpretation, daß die Informationstechnologien selbst die unmittelbare und entscheidende Ursache für diese Vorgänge sind.

Genau aus diesem Grunde ist es wichtig, nicht nur die auf der Oberfläche erscheinenden sozialen Wirkungen für das Wesen der Sache selbst zu nehmen, sondern die dahinter liegenden gesellschaftlichen Gründe und Ursachen für diese Erscheinungen zu analysieren. Die gesellschaftlichen Grundstrukturen und Grundprozesse werden durch den Einsatz der Informationstechnologien nicht neu formiert oder determiniert, sondern nur in ihrem Wirkungsmechanismus verstärkt und beschleunigt. Die Informationstechnologien schaffen keine prinzipiell neuen gesellschaftlichen Probleme, sondern führen zur schnelleren Entfaltung der in ihnen wurzelnden Entwicklungstendenzen. Informationstechnologien sinddaher Tendenzverstärker allgemeiner gesell-

schaftlicher Entwicklungsprozesse und nicht Ursachen für eine prinzipielle Veränderung unserer Beurteilung des Wesens der Gesellschaft selbst.

Die Unterscheidung zwischen sozial und gesellschaftlich widerspiegelt sich bspw. auch in der Redeweise von einer Wirtschafts- und Sozialpolitik, die zwar eine Einheit bilden, aber eben nicht identisch sind. Die Wirtschaftspolitik dient der schnellen Entwicklung der Produktivkräfte unter Nutzung aller Ergebnisse der wissenschaftlich-technischen Revolution, während die Sozialpolitik die Art und Weise der Produktivitätsentwicklung determiniert. Da die Arbeitskraft die wichtigste Produktivkraft ist, alles was die Menschen tun durch ihren Kopf hindurch muß, kommt es eben darauf an, die Menschen selbst als Produzenten dazu zu befähigen, die notwendigen Wandlungsprozesse in der Produktion so zu realisieren, daß die Veränderung der Arbeits- und Lebensbedingungen akzeptiert wird und sogar Triebkräfte für die Realisierung solcher Wandlungsprozesse freigesetzt werden. Die gesellschaftlichen Wirkungen der EDV-Anwendung bestehen also darin, daß die bestehenden Gesellschaftsstrukturen sichtbarer ausgeprägt werden, das heißt konkret, daß sich im Kapitalismus der Widerspruch zwischen Kapital und Arbeit weiter verschärft und im Sozialismus die prinzipiellen Vorzüge für eine gesamtgesellschaftliche Organisation der Produktion immer umfassender durchsetzen. Diese Kennzeichnung bedeutet jedoch nicht, daß der Einsatz der Informationstechnologien automatisch zu solchen gesellschaftlichen Wirkungen führt, weil ja die Umsetzung der mit den Informationstechnologien verbundenen Erfordernisse immer wieder neu Entscheidungen über Gestaltungsprinzipien und Einführungsprozesse auf die Tagesordnung setzt.

Wir hatten vorher zu zeigen versucht, daß jeder Entwicklungsprozeß irreversibel und ambivalent ist. Das bedeutet, daß die Wirkungen des Einsatzes von Informationstechnologien und der Robotertechnik in der materiellen Produktion zwangsläufig zu Veränderungen von Arbeitsinhalten führt, die nicht nur positiv bewertet werden können. Es sind ja nicht nur die körperlich und psychisch unangenehmen Teile des menschlichen Arbeitsprozesses, die fundamental und irreversibel verändert werden, sondern es

ändern sich auch Identifikationen der Menschen mit ihrer Qualifikation und ihren Verantwortungsbereichen im Arbeitsprozeß. Es wäre naiv zu glauben, daß nach diesen Veränderungen nur schöpferische und persönlichkeitsstimulierende Tätigkeiten übrig bleiben. Das Verhältnis von schöpferischen und nichtschöpferischen Tätigkeiten wird lediglich verschoben, nicht prinzipiell aufgehoben. Das ist auch ein wesentlicher Grund dafür, daß die Neuerungen nicht nur emphatisch begrüßt, sondern zugleich auch abgelehnt werden, wenn die mit der Einführung der Informationstechnologien verbundenen neuen Anforderungen als weniger interessant empfunden werden als der bisherige Arbeitsprozeß. Man muß daher jeweils die Gesamtheit der Bedingungen und Faktoren in Rechnung stellen, die vor und nach Einführung der Informationstechnologien die Arbeits- und Lebensbedingungen bestimmen.

Aus dem bisher Gesagtem geht hervor, daß die Analyse eines weiteren Wirkungsaspekts notwendig wird, um die sich vollziehenden Wandlungsprozesse umfassend charakterisieren zu können. Ich meine hierbei die Organisationsstrukturen. Der Organisationsaspekt ist nicht nur ein Bestandteil des Arbeitsprozesses, sondern durchzieht alle Formen des gesellschaftlichen Lebens, wie Politik, Kultur und Bildungswesen, Gesundheitswesen, Sport und Erholung und viele andere Bereiche.
Das Niveau einer gesellschaftlichen Entwicklung zeigt sich nicht nur am Stande von Wissenschaft und Technik und ökonomischen Produktionsziffern, sondern auch an den damit verbundenen Strukturen und Mechanismen der Organisation aller Arten von Lebensprozessen. Insbesondere die Informationstechnologien führten zu einer Hervorhebung und einer relativen Verselbständigung des Organisationsaspekts, weil insbesondere die Organisation selbst der unmittelbare Gegenstand dieser Technologien geworden ist. Das zeigt sich schon äußerlich darin, daß gut durchdachte Organisationslösungen von ihrem konkreten Inhalt abhebbar gemacht werden, um auf diese Weise automatisierte Informationsverarbeitungssysteme übertragen und nachnutzen zu können. Es spielt unter diesem Gesichtspunkt eben keine Rolle, ob bspw. im Gesundheitswesen Medikamente bestellt, verwaltet und verteilt werden müssen oder im Rahmen von Handel und Versorgung Bestandhaltungs- und Belieferungsprobleme für Warenhäuser zu

lösen sind.
Das Hauptproblem der Organisation ist die Beherrschbarkeit von Planungs- und Koordinierungsprozessen. Jeder einzelne dieser ineinandergreifenden Teilprozesse ist durch eine Menge von Informationen charakterisiert. Je mehr es gelingt, die Prozesse möglichst genau durch die für sie wesentlichen Informationen zu repräsentieren, um so besser gelingt es, mit diesen Abbildern wirklicher Vorgänge informatorisch zu arbeiten, Organisationsprobleme zu analysieren und eingreifende Organisationsentscheidungen zu treffen. Da nun die Computertechnik nur formalisierte Informationen zu verarbeiten gestattet, werden die Teilprozesse selbst vorwiegend durch Datensätze gekennzeichnet. Es wird hierbei stillschweigend unterstellt, daß alle Organisationsphänomene auf diese Weise am zweckmäßigsten und effektivsten erfaßt werden können. Man spricht in diesem Zusammenhang auch von der Modellierung der durch die Organisation zu gestaltenden Phänomene. Auch der Mensch selbst als inhärentes Moment jeder Organisation wird hierbei leicht zu einem Datensatz. Viele bürgerliche Wirkungsforscher beklagen daher den Substanzverlust, der als ambivalente Kehrseits des Organisationsfortschritts unweigerlich in Kauf zu nehmen ist. Der Computer wird nicht nur zu einem Hilfsmittel bei der Bewältigung von Organisationsproblemen, sondern er avanciert mehr und mehr zu einem Modell perfekter Organisation. Die Verkehrung einer Mittel/Zweck-Beziehung ist in der marxistischen Philosophie an vielen Entwicklungsproblemen anschaulich demonstriert worden. Die Informationstechnologien sind gewissermaßen die moderne Form einer Reflexion über Mittel und Zweck, bezogen auf die in großen Dimensionen vor sich gehenden theoretischen und praktischen Veränderungen unserer Auffassungen über Organisation und ihre Handhabung.
Auch hier zeigt sich wieder die Ambivalenz jeder Art von Fortschritt, hier bezogen auf den Organisationsaspekt. Meines Erachtens haben sich die Gesellschaftswissenschaften unserer Tage noch zu wenig mit diesen aktuellen Problemen beschäftigt und allgemein anerkannte Lösungen vorgelegt. Wenn es wahr ist, daß Organisationsfortschritt mit Formalisierung der Organisationsstrukturen zusammenhängt, dann darf wenigstens vermutet werden, daß die sich daraus ergebenden Konsequenzen nicht unproblematisch sind und weitere Analysen dringend erforderlich machen. In

der bürgerlichen Wirkungsforschung wird dieser Aspekt des Organisationsfortschritts mit wachsender Bürokratisierung gesellschaftlicher Lebensprozesse in Zusammenhang gebracht und hieraus eine neue Quelle für menschliche Selbstentfremdung abgeleitet. Sicherlich ist dieser Kurzschluß nicht unbesehen zu akzeptieren, wie aber der Organisationsfortschritt umfassend aus marxistisch-leninistischer Sicht zu kennzeichnen wäre, ist das meines Erachtens noch nicht voll gelöste Problem unserer Tage.

Als einen weiteren Aspekt der Wirkung von Informationstechnologien hatten wir den Lebensprozeß erwähnt. Der Lebensprozeß kennzeichnet die sogenannte Lebensweise. Gerade in den letzten Jahren nehmen die Forschungen über die theoretische Bestimmung des Phänomens "Lebensweise" und die Kennzeichnung ihrer Faktoren zu. Man spricht von der Lebensweise im Kapitalismus und im Sozialismus. Verbunden mit der Lebensweise ist das qualitative und quantitative Lebensniveau, gewissermaßen all das, was das Leben lebenswert macht. Ich will hier gar nicht erst versuchen, eine präzise Bestimmung des gesellschaftlichen Phänomens "Lebensweise" zu geben.
Für unsere Zweoke reicht eine relativ vage bleibende Skizzierung des Gebietes, zu dem jeder einen praktischen und anschaulichen Zugang hat.
Wie verändert nun der Einsatz der Informationstechnologien unsere Lebensweise?
Hervorstechend ist hier vor allem der schnelle Ausbau der Kommunikationsdienste. Die Kommunikation ist nicht nur ein Hilfsmittel für die Koordinierung von Tätigkeiten, für die Lern-, Qualifizierungs- und Orientierungsprozesse in unserem Leben, sondern selbst eine menschliche Tätigkeit. Die Kommunikation ist geistige Vergegenständlichung von Ideen und Absichten, von Zielen, Urteilen und Bewertungen. Sie ist primär das Aktionsfeld, in dem Subjekte aufeinander wirken und sich wechselseitig beeinflussen und verändern. Auch das menschliche Selbstverständnis benutzt die kommunikativ gewordenen Bezeichnungen für Erkenntnisse und Bewertungen. Die Kommunikation ist gewissermaßen unsere unmittelbare Lebenssphäre.
Gerade deshalb ist es so wichtig, die EDV-Rationalisierungen der Kommunikation richtig einzuordnen. Ihre Ambivalenz liegt in

der Gefahr, Kommunikation auf das maschinell Machbare zu reduzieren und darüber hinaus als Fortschritt zu feiern. Auch die vorsichtigere Interpretation, nämlich die Leistungsfähigkeit der Kommunikation durch den Computereinsatz zu erhöhen, ist schon deshalb nicht ausreichend, weil Inhalt und Funktion der menschlichen Kommunikation unanalysiert bleiben. Kommunikation ist primär nicht - wie in der EDV-Literatur meist zu lesen ist - Informationsaustausch, sondern Austausch von Meinungen, Gedanken und Argumentationen. Damit wird das weite Feld der Beziehungen zwischen Informationen und Werten berührt. Meinungen sind gewertete Informationen, "meine" Sicht der Sachverhalte. Bei der Argumentation fungieren die Werte sogar als Ausgangspunkt und Ziel der Kettung von Aussagen. Genauso unabweisbar ist die Wertproblematik für das Verständnis von Entscheidungsprozessen, die man eben nur dann mit ruhigem Gewissen einem Computer überantworten kann, wenn man zuvor den Menschen zu einem verdinglichten Sachverhalt gemacht hat. Hier kann man geradezu mit Händen greifen, wie die Grundtendenz der kapitalistischen Deformierung des Menschen zu einem seelenlosen Rädchen im Profitmechanismus durch den Computer praktisch und durch eine entsprechend orientierte computer-science theoretisch verstärkt wird und gewissermaßen nachträglich eine als "wissenschaftlich" deklarierte Begründung erfährt.

An der Kommunikations-Problematik scheiden sich meines Erachtens die Geister, die die modernen Informationstechnologien heraufbeschworen haben, viel fundamentaler als an anderen EDV-Anwendungen und ihren Auswirkungen. Gerade auf diesem Gebiet muß gedankenlosen Vereinfachungen viel energischer widersprochen werden, um die Computer in die Schranken zu verweisen, damit die Menschen sich ihrer eigenen Rolle besser bewußt werden können, damit ein sinnvoller Gebrauch der neuen Technik möglich wird. Ich möchte daher das Wesen der menschlichen Kommunikation noch einmal kurz umreißen, damit es durch übersteigerte Computer-Euphorie nicht in Vergessenheit gerät:

Es genügt nicht, beim Sender und Empfänger "ein gemeinsames Alphabet" und eine Übereinstimmung in den Wortbedeutungen (Begriffen) zu fordern, um die Kommunikation von Meinungen und

Gedanken erfolgreich zu gestalten. Die Modelle, die die kommunizierenden Partner voneinander haben, sind Subjektmodelle, Antizipationen der Persönlichkeitsstruktur, an die die Ideen gerichtet sind, an der eine Wirkung erzielt werden soll.

In der Kommunikation treten die Menschen daher einander als unverwechselbare Individuen gegenüber, nicht austauschbar. Daher nimmt diese Kommunikation selbst einen individuellen Zug an und ist im engeren Sinne des Wortes zwischenmenschlich. Sie dient entweder vorwiegend der Einflußnahme auf die Ideen des anderen oder ist eine Offenbarung der eigenen Ideen in der Hoffnung, daß der andere helfen möge, nicht mehr stimmige Informationserzeugungsprozesse aus seiner Sicht zu analysieren und damit Hilfe zu geben.

Ein wesentlicher Grundzug der Ideen-Kommunikation ist daher das möglichst komplexe Verständnis für den anderen, die einfühlsame Teilnahme an seiner Subjektivität. Dieses Verständnis ist um so tiefgreifender und umfassender, je ähnlicher die vom anderen empfundenen Konflikte und Interpretationen den eigenen Erfahrungen sind, die uns geprägt haben, je mehr also derartige Situationen durchlebt wurden.

Es sind nicht die Kenntnisse und Erkenntnisse selbst, die bei einer Ideen-Kommunikation ausgetauscht werden, obwohl davon vordergründig die Rede ist, sondern die Sicht der Problemzusammenhänge, aus der diese Erkenntnisse erwachsen.

Ein durch viele praktische Beispiele belegbares Phänomen der Ideen-Kommunikation ist der Sachverhalt, daß mitunter mit wenigen Worten sehr viel oder umgekehrt mit sehr viel Worten sehr wenig Ideen übertragen werden können. Die Informationsbedeutungen, die Münzen, in denen wir zahlen, sind nur aktivierende Werkzeuge für Eingriffe an den fremden Ideen. Sie sind nur Mittel, nicht Zweck der Kommunikation.

Während in der objektiven Welt die Kräfte die gegenseitigen Beeinflussungen realisieren und einen Gesamtzusammenhang der Dinge und Erscheinungen konstituieren, stehen dem Subjekt für seine Wirkungen auf andere Subjekte nur die Möglichkeiten der kom-

munikativen Einflußnahme auf deren Struktur zur Verfügung. Kommunikation ist gewissermaßen das Kraftfeld, in dem sich Subjekte verändern, in dem sie existieren und sich bewähren.

Die Veränderung der technischen Grundlagen und technischen Möglichkeiten für die kommunikative Erreichbarkeit verändert nachhaltig die Lebensbedingungen der Menschen. In erster Linie ist hier an die neue Generation von Massenmedien zu denken. Die Auswirkungen auf die Persönlichkeitsentwicklung sind keineswegs restlos aufgeklärt. Es gibt viele Langzeitforschungen, die sich diesem Problemkreis zugewandt haben.

Die Aufgabe, vor der wir theoretisch und praktisch stehen, ist die Verbindung von wachsender Objektivierung einer immer größer werdenden Zahl von gesellschaftlichen Prozessen durch den Einsatz der Informationstechnologie mit einer schritthaltenden Subjektivierung, d.h. Subjektwerdung des Menschen. Die Entfaltung der Schöpferkräfte des Menschen ist und bleibt das Bewertungsmaß für den Fortschritt. Computer sind Vergegenständlichungen dieser Kräfte, nicht aber sie selbst. Sie müssen für uns handhabbar gemacht werden, nicht wir für sie. Der Mensch ist niemals ein Störfaktor perfekter Organisation. Sorgen wir gemeinsam dafür, daß der Computer kein Störfaktor für uns und unsere Entwicklung wird!

Literaturhinweise:

W. Steinmüller "Datenverkehrsrecht oder Informationelles Selbstbestimmungsrecht des Bürgers? - Tendenzen und Probleme der Informationsautomation" aus "Mensch und Computer": Zur Kontroverse der ökonom. u. gesellschaftl. Auswirkungen der EDV/ hrsg. von Hans Robert Hansen ... München, Wien: Oldenbourg, 1979

Verfasser:
Prof. Dr. Bodo Wenzlaff
Humboldt-Universität zu Berlin
Sektion Wissenschaftstheorie und -organisation
1086 Berlin, August-Bebel-Platz

Bilgrim, Jürgen

# Der EDV-Benutzer - Versuch einer Beschreibung und Definition

Zunehmend muß sich der Mensch auseinandersetzen mit verschiedenen Formen der elektronischen Rechentechnik: mit Rechnern, Terminals, Mikroprozessoren und Bürocomputern. Dabei wird erwartet, daß er diese Technik und entsprechende Nutzungstechnologien sinnvoll beherrscht. Nur dann können die Computersysteme ihre volle Effizienz erreichen.
Der sich verstärkende Übergang zur dezentralisierten Datenverarbeitung - ein Ende dieser Entwicklung ist derzeitig überhaupt nicht zu erkennen - forciert diese Aussagen.
Die bisherige Arbeitstätigkeit von Menschen, ihr Inhalt, werden somit durch Informat-ionstechnologien verändert oder bereichert oder sogar verdrängt. Darin liegt die eigentliche Problematik der EDV mit ihrem Grundcharakter als massiv arbeitseinsparende Technologie.
Vom Menschen als Benutzer werden erwartet: Verständnis, Anpassung und Engagement gegenüber den Informationstechnologien, wobei die bisherige Entwicklung der Computerisierung in den kapitalistischen Ländern zeigt, daß das außerordentlich schwierig und bezüglich sozialer Auswirkungen in gesellschaftlichen Dimensionen dort kaum beherrschbar ist.
Aufgrund der zunehmenden Automatisierung ergibt sich auch für uns die Notwendigkeit, sich mit derartigen Fragen zu beschäftigen, um unerwünschte soziale Auswirkungen und Hemmnisse zu vermeiden.

## 1. Bisherige Positionen des Benutzers in der EDV

Es lassen sich in der bisherigen Entwicklung der EDV unterschiedliche Positionen des EDV-Benutzers feststellen:

1. Etappe (um 1955 bis Mitte der 60'er Jahre):
   Übergang vom reinen Computing auch zur Zeichenkettenverarbeitung und Time-Sharing-Systemen. Entwicklung neuer Programmiersprachen (FORTRAN 1954, ALGOL 1960). Sie führen zu:
   - der Erweiterung der EDV vom reinen Hardwareverständnis

zum Software-Hardware-Komplex, insbesondere durch Einbeziehung von Software und Information (Datenbanken)
- ersten Problemen der Mensch-Rechnereffektivität und daraus resultierendem Interesse an vergleichbaren Experimenten und Studien(vgl. SACKMAN / 1 /)

2.Etappe (Mitte der 60'er Jahre bis um 1975):
Enorme technische Weiterentwicklung (Verkleinerung, Verbilligung, Massenhaftigkeit, Dezentralisierung), z.B. 1965: IBM 360, PDP 8; 1966: Displayterminal; 1971: ARPA - Rechnernetz, LSI-Mikroprozessoren.
Es entsteht eine wachsende Akzentuierung von Mensch-Maschine-Systemen, und ein vom Computer determiniertes Denken entwikkelt sich, das alles für formalisierbar hält( vgl. NEWELL und SIMON / 2 /.
Es werden realisiert ergonomische Untersuchungen von MMS, da da der Mensch als Benutzer zum begrenzenden Faktor wird (Adaptionszwang!), wobei aber allgemein die Nutzeruntersuchungen nicht Schritt halten (NICKERSON / 3 /).

3.Etappe ( seit der 1.Hälfte der 70'er Jahre):
Multidimensionalität des Leistungsvermögens und Anwendung der EDV als Informationstechnologie.
Sie führen zur zunehmenden Kritik infolge wachsender sozialer Auswirkungen, verbunden mit der Einführung der Mikroelektronik. Kritiker sprechen vom Neotaylorismus (vgl. u.a. WEIZENBAUM / 4 /). Die Abwehr negativer sozialer Auswirkungen erscheint als wichtigste Motivation in den kapitalistischen Ländern.
Parallel zu dieser Entwicklung entsteht die sogenannte Wirkungsforschung, vor allem in den skandinav-ischen Ländern (vgl. BJÖRN-ANDERSEN und BORUM / 5 /), gekennzeichnet durch die Einbeziehung des Nutzers in die Systemauffassung und zwar zunächst als passive Komponente und jetzt als Mitgestalter.

Es wird somit der Mensch zunehmend als EDV-Benutzer interessant
- eine arbeitsmäßige Funktion und Rolle, die er überwiegend als Nicht-EDV-Fachmann ausübt, und die künftig auch in der privaten Sphäre wahrzunehmen ist.

Damit werden zwei Gesichtspunkte gravierend:

- die zeitlich unterschiedlich determinierte und ausgeübte Rolle als EDV-Benutzer neben einer Vielzahl anderer Rollenfunktionen,
- die Ausübung dieser Rolle als Nichtfachmann der EDV.

## 2. Zur bisherigen Beschreibung des EDV-Benutzers

Analysiert man den EDV-Benutzer im bisherigen EDV-Schrifttum, so zeigt es sich, daß eine ausreichende, umfassende Definition über ihn nicht existiert. Es gibt eine Vielzahl von fragmentarischen Ansätzen, z.B. bei MARTIN / 6 /, U.E.FISCHER / 7 /, und Begriffen wie aktive und passive Nutzer, first-time Nutzer und first-come Nutzer, Problemlöser u.a., die aber mehr die vernachlässigte Rolle des Menschen als Benutzer in der Informationstechnologie betonen als eine erschöpfende Auskunft vermitteln. Schon 1970 sprach H.SACKMAN vom vergessenen Menschen in der EDV / 1 /.

In der Fachliteratur der kapitalistischen Länder nimmt seit einigen Jahren der EDV-Benutzer eine wachsende, signifikante Position ein (vgl. auch: MUMFORD und SACKMAN / 8 /; HANSEN, SCHRÖDER und WEIHE / 9 /), wobei an ihm, resultierend aus den gesellschaftlich unkontrollierten sozialen Auswirkungen, zunehmend die "Negativität" der Computerisierung betont wird - ohne die eigentlichen gesellschaftlichen Hintergründe für diese Phänomene explizit zu benennen.

Demgegenüber soll es unser Anliegen sein, den EDV-Benutzer unter dem Gesichtspunkt der Produktivitäts- und Kreativitätsverbesserung für die EDV-Nutzung und damit für den zugrundeliegenden Arbeitsprozeß zu untersuchen, und somit den positiven Beitrag der EDV zur Entwicklung der Produktivkräfte aber auch des menschlichen Denkens in Abhängigkeit von der gesellschaftlichen Determiniertheit hervorzuheben und zu forcieren.

Derzeitig lassen sich mindestens 6 unterschiedliche Positionen bei der Bestimmung oder Beschreibung des EDV-Benutzers nachweisen, denen unterschiedliche Standpunkte aber auch verschiedene wissenschaftliche Disziplinen zugrundeliegen:

1. Die Auffassung des EDV-Benutzers als Mensch, als besonderer Verhaltenstyp, charakterisiert durch bestimmte psychische Eigenschaften - und damit als ein Begriff der Psychologie

(WEINBERG /10/ spricht z.B. vom abgesonderten Typ des Programmierers; vgl. auch die interessante Arbeit von SHNEIDERMAN / 11 /).

2. Die Auffassung des EDV-Benutzers als eine Identifikationsnummer in einer Warteschlange von Rechenaufträgen, d.h., als Element von bedienungstheoretischen Modellansätzen des Jobdurchsatzes (vgl. u.a. BOIES / 12 /).
3. Die Auffassung des EDV-Benutzers als Element der Funktionseinheit Mensch-Maschine-System, d.h., als Teilleistungskomponente eines Gesamtsystems (vgl. Mc CORMICK / 13 /), d.h., vorwiegend als kybernetische Interpretation unter informationeller Betonung (Kommunikationsproblem!), oder auch unter arbeitswissenschaftlichen Gesichtspunkten (NEUMANN und TIMPE / 14 /).
4. Die Auffassung des EDV-Benutzers als Element einer Gesamtheit, d.h., einer Gruppe mit bestimmten, übereinstimmenden Merkmalen wie z.B. der medizinische EDV-Benutzer, der Systemprogrammierer usw.. Diese Auffassung ist weit verbreitet und dient meist der Systematisierung von Nutzerkategorien oder Nutzungsanforderungen.
5. Die Auffassung des EDV-Benutzers als Element des Nutzersystems, d.h., nur in Verbindung mit seiner EDV-Aufgabe und damit aus der Sicht des Rechnerbetreibers und -entwicklers (vgl. PILGRIM / 15 / und angenähert bei EASON / 16 /).
6. Die Auffassung des EDV-Benutzers als Initiator und Träger der Arbeitstätigkeit "EDV-Nutzung", d.h., als eine Kategorie der Informatik, die von uns vertreten wird.

## 3. Das Nutzersystem in der EDV

Unter Nutzersystem wird von uns verstanden das funktionelle Zusammenwirken des EDV-Benutzers mit seiner arbeitsmäßigen Umgebung in bezug auf die Inanspruchnahme von EDV-Leistungen bei der Bearbeitung seiner EDV-Aufgabe. Dabei besteht ein Nutzersystem ($N_S$) mindestens aus folgenden Elementen, die diese Repräsentanz widerspiegeln:

- EDV-Benutzeraktivitäten ($B_A$), d.h., der arbeitsmäßig eigene Anteil des Benutzers an der Realisierung der EDV-Aufgabe (z.B. nur die Aufgabe vorgeben als Auftrag oder: nur Ergeb-

nisse berechnen und auswerten oder: gesamte Programmentwicklung selbst durchführen ),

- die Motivation und Einstellung zur EDV-Nutzung (als rekursive Grössen aus dem Gruppeneinfluß, der insbesondere als Gruppenziel die Problemlösung und nicht die EDV-Aufgabe verfolgt), ($B_M$),
- die EDV-Qualifikation des EDV-Benutzers (persönliche Erfahrungen, Übungen und Lehrgänge), ( $B_Q$),
- das Niveau der EDV-Aufgabe als Bearbeitungsprozeß, ( $E_A$),
- die rechentechnischen Anforderungen der EDV-Aufgabe zwecks ihrer Bearbeitung und Lösung. ( $E_R$).

Es ergibt sich dann folgendes Strukturmodell eines Nutzersystems (Abb. 1 ):

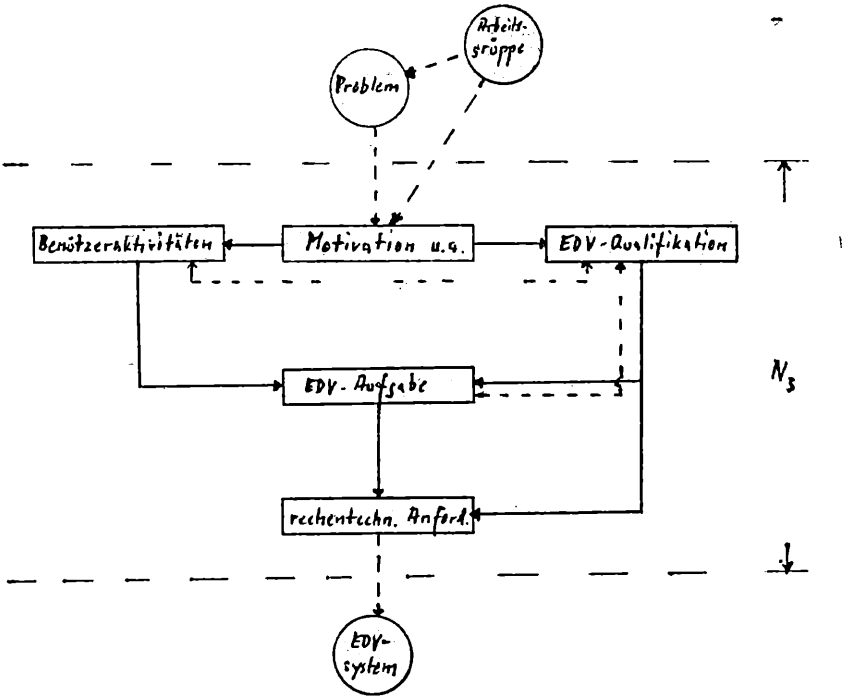



Abb. 1 Struktur von $N_S$

Es zeigt sich eine bestimmte Polarität in diesem Modell in bezug auf

- Motivation und Einstellung zur EDV und
- rechentechnische Anforderungen,

die gleichzeitig als $N_s$ - Elemente auch die Schnittstellen zur Umgebung beschreiben. Sie sollten deshalb besonders das Objekt von Nutzeruntersuchungen bilden.

Aus der Sicht eines Rechenzentrums als Betreiber von Rechentechnik hat ein Nutzersystem den Charakter eines Systems, das sich laufend modifiziert

- im Ergebnis von Veränderungen der Einflußgrössen einschließlich der Elemente selbst,
- in Anpassungen an die EDV-Nutzung und daraus resultierender Anforderungen.

Daraus ergibt sich die Notwendigkeit der möglichst quantitativen Bestimmung des Zustandes als auch der Entwicklung eines Nutzersystems. Der Zustand eines $N_s$ läßt sich durch einen Vektor beschreiben, wenn für den Index i ein aktueller Wert für die Elementmerkmale oder Indikatoren von $N_s$ eingesetzt werden kann. Die Merkmale erhalten damit bewertet den Charakter von Variablen.

Beispiel:

Rechentechnische Anforderungen

Indikator:

| Kontinuität der Inanspruchnahme des Rechners | Index i (i = 1...5) |
|---|---|
| - kontinuierlich, regelmäßig | 1 |
| - kontinuierlich, mit Unterbrechungen | 2 |
| - häufig | 3 |
| - selten, unregelmäßig | 4 |
| - sehr selten | 5 |

weitere Indikatoren:

z.B. Verweilzeit des Rechenauftrages,
benötigte Dateien (Umfang).

Aus der Sicht des Betreibers und Entwicklers von elektronischer Rechentechnik wird dann der Zustand von $N_s$ identisch mit einem bestimmten, erreichten Kompliziertheitsgrad von $N_s$ , den wir mit $K_N$ bezeichnen.
Es gilt:

$N_s = ( B_A, B_M, B_Q, E_A, E_R )$

$B_A = a_i, b_i$

$B_M = c_i, d_i, e_i$

.

$K_N = a_i + b_i + c_i + d_i + e_i \ldots + n_i$

Mit dieser Ableitung wird ersichtlich, daß die Entwicklung eines $N_s$ aufgefaßt wird als Entwicklung seiner Kompliziertheit $K_N$ , die aber immer bewertet wird vom Standpunkt des Betreibers oder Entwicklers. Damit zeigt sich die Kompliziertheit $K_N$ als Prozeß im Zeitabschnitt $\Delta_t$ , den man als "Lebensdauer" eines $N_s$ bezeichnen kann. Das bedeutet, daß ein Benutzer mit seiner EDV-Aufgabe und ihrem Bearbeitungsprozeß mit einer variablen Kompliziertheit K für einen bestimmten, oft vorher nicht abschätzbaren Zeitabschnitt für ein Rechenzentrum oder einen Computer als sogenanntes Nutzersystem existent und nachweisbar ist.
Die Entwicklung von $N_s$ und damit seiner Kompliziertheit ist als stochastischer Prozeß zu bewerten, wobei die Elemente von $N_s$ einer unterschiedlichen Dynamik unterliegen (Abb. 2 ).
Der grafische Ansatz in Abb. 2 ist grob und repräsentiert nur die Tendenz. In der Praxis zeigt die Kompliziertheit der einzelnen Elemente eine nichtlineare Entwicklung, die zusätzlich noch durch einen diskontinuierlichen Verlauf oder Wirksamkeit der Elemente und ihrer Indikatoren beeinflußt wird.
So bleibt der Zuwachs an Kompliziertheit bei $B_A$ (Benutzeraktivitäten) sehr gering bzw. gleich Null: der Benutzer realisiert bei einer konkreten EDV-Aufgabe allgemein das gleiche Niveau an Aktivitäten . $B_M$ (Motivation und Einstellung) ver-

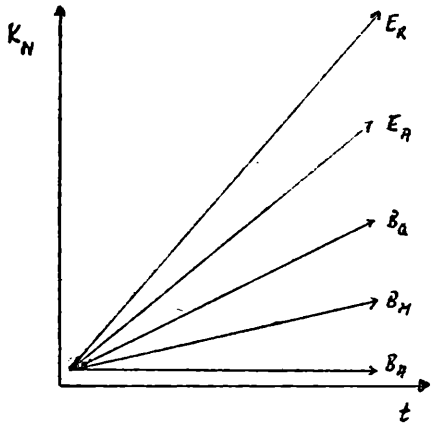



Abb. 2 Entwicklung der Elemente von $N_s$

ändert sich nur langsam aber dann sprunghaft im Sinne einer Stufenfunktion. Motivation und Einstellung, basierend auf einem überwiegend problemorientierten, fachlich und sozial bedingten Einstellungshintergrund sind in kurzen Zeiträumen kaum zu verändern (Ausnahmen: langzeitige Nutzersysteme in Gestalt von Programmsystemen!).

Der Kompliziertheitszuwachs bei der EDV-Aufgabe ($E_A$) ist typisch für langzeitige oder langlebige Nutzersysteme im Bereich der Forschung. In Abhängigkeit vom problemorientierten Erkenntniszuwachs entwickeln sich aus kleinen Computerprogrammen über wiederholte Korrekturen und Verbesserungen große und komplizierte Programmsysteme, oft mit einer Vielzahl von Unterprogrammen.

Am stärksten dürften sich die rechentechnischen Anforderungen ( $E_R$ ) entwickeln - in Abhängigkeit vom Niveau der Entwicklung der EDV-Aufgabe ($E_A$) und vom Zuwachs an EDV-Kenntnissen und -Erfahrungen. Gerade bei Erstnutzern ist dieser Zuwachs besonders ausgeprägt, wobei häufig nutzerseitige Überforderungen bezüglich rechentechnischer Anforderungen vorherrschen.

Die Entwicklung eines Nutzersystems wird von verschiedenen

Ursachen bestimmt:

1. Es existiert eine unterschiedliche Entwicklungsintensität und -tendenz der Elemente von $N_S$.
2. Die Elemente von $N_S$ partizipieren zeitlich nicht parallel an der Entwicklung der Kompliziertheit $K_N$ von $N_S$.
3. Änderung der Aufwandsrelation Entwicklung : Anwendung(Routinenutzung), z.B. bei einer Programmentwicklung. In Abhängigkeit vom Bearbeitungsfortschritt erhöht sich allgemein der Routineanteil (vgl. Abb. 3 ).
4. Unkontinuierliche Arbeiten an der Programmentwicklung, d.h., Realisierung von EDV-fremden Tätigkeiten neben der EDV-Aufgabe und somit Vorhandensein sogenannter Totzeiten für die EDV-Nutzung.

Abb. 3 zeigt eine modellartige Vorstellung der prinzipiellen Entwicklung eines $N_S$. Typisch ist dabei die Verringerung des Zuwachses von $K_N$ mit wachsendem t .

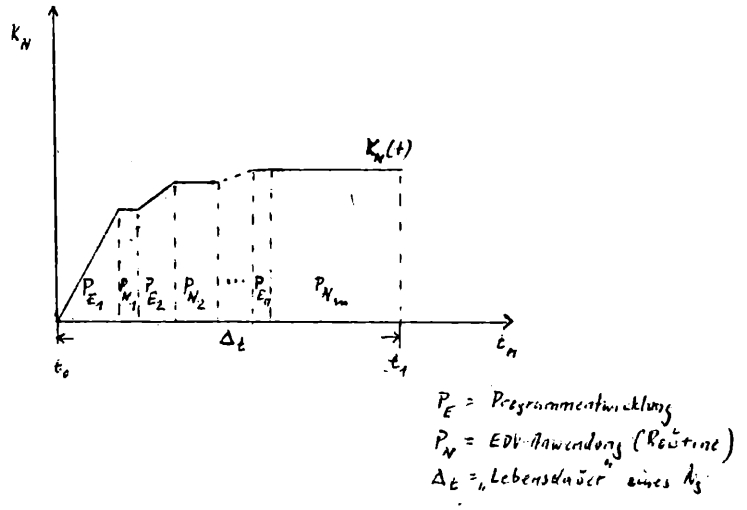



Abb. 3 Entwicklung eines $N_S$ (vereinfacht)

Alle 6 o.g. Auffassungen über den EDV-Benutzer, mit Ausnahme der letzten, sind partiell und gezielt anwendbar, beschreiben aber nicht umfassend den EDV-Benutzer.
Nachfolgend soll die letztgenannte Auffassung betrachtet werden, in der der EDV-Benutzer als Träger der EDV-Nutzung aber auch als soziale Komponente der Informationstechnologie verstanden wird. Mit dieser Interpretation erhält der EDV-Benutzer den Charakter einer " 3.Dimension" in der Informatik - neben der Hardware und Software!
Es wird dann erforderlich, 2 Schwerpunkte vordergründig zu untersuchen:

1. Die Analyse der arbeitsmäßigen Einbindung des EDV-Benutzers in die EDV, d.h., die Untersuchung der Arbeitstätigkeit "EDV-Nutzung".
2. Die Analyse der arbeitsmäßigen Umgebung des Benutzers, d.h., die soziale Wechselwirkung während der EDV-Nutzung und mittels der Informationstechnologie.

## 4. Die EDV-Nutzung als Arbeitstätigkeit

Der EDV-Nutzung als Arbeitstätigkeit liegen allgemein folgende Aktivitätengruppen zugrunde (Abb. 4 ):

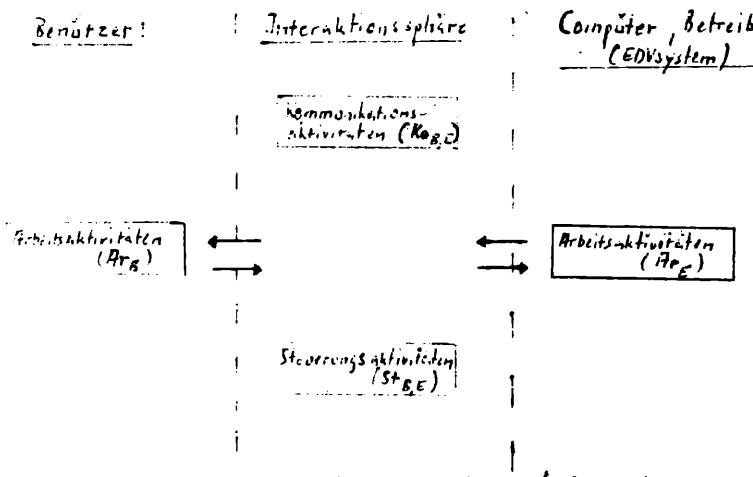



Abb. 4 Modell der Arbeitstätigkeit „EDV-Nutzung" (vereinfacht)

1. Arbeitsaktivitäten
als eigentliche Tätigkeitsn eines bestimmten Trägers, die somit immer trägerbezogen und die Erfüllung der EDV-Aufgabe unmittelbar verwirklichen, d.h., finalorientiert am Endprodukt (z.B. Rechenprogramm) sind (Beispiel: Schreiben von Programmanweisungen).

2. Steuerungsaktivitäten
zwecks Steuerung des Ablaufs der EDV-Nutzung. Sie zeigen sich als Resultat von Kommunikationsaktivitäten und bewirken gezielte Eingriffe in Handlungsabläufe (Prozesse) (Beispiel: Zusammenstellen von Jobsteuerkarten für einen Rechenauftrag.

3. Kommunikationsaktivitäten
mit dem Ziel einer Informierung des Veranlassers bzw. des Empfängers u.a. für Steuerungsprozesse (bzw. -aktivitäten) Beispiel: Fehlermitteilungen durch das System).

Man kann nun folgende Beziehung herstellen:

Arbeitstätigkeit EDV-Nutzung ( $Ar_{EDV}$)

$$= Ar_B + Ko_B + St_B + Ar_E + Ko_E + St_E;$$

$$Ko_B + St_B = D_N$$

$$Ko_E + St_E = D_E$$

Alle 3 Arten von Aktivitäten partizipieren träger- und umfangsmäßig, inhaltlich und zeitlich unterschiedlich in jeder EDV-Nutzung. Das betont in Abhängigkeit vom Typ der EDV-Nutzung (z.B. Datenerfassung oder Programmentwicklung) die mögliche und oft tatsächliche Individualität von EDV-Nutzungen, die dabei vor allem vom EDV-Benutzer bestimmt und getragen wird.

Hier liegt die eigentliche Problematik der Dialektik von Flexibilität und Starrheit im EDV-Einsatz, die sich in einem Pro oder Contra der EDV-Nutzungsfreundlichkeit in der Praxis niederschlägt, über Akzeptanz oder Ablehnung von Informationssystemen durch den Benutzer sichtbar wird. Hier wird über die

tatsächliche Effektivität der EDV-Anwendung entschieden. Andererseits befinden sich auf diesem Gebiet auch die Reserven bezüglich Aufwertung des EDV-Benutzers und seines Einsatzes als Produktivitäts- und Kreativitätsfaktor.

Kommunikations- und Steuerungsaktivitäten charakterisieren die eigentliche und notwendige Wechselwirkung des Benutzers mit der EDV-bezogenen arbeitsmäßigen Umgebung, z.B. dem Computer; sie zeigen externes Verhalten des Benutzers an. Insofern wird die o.g. Individualität oder auch vorhandene Isolierung des Menschen bei einer Programmentwicklung durch diese Aktivitätengruppen, insbesondere durch Kommunikationsbeziehungen, durchbrochen und kann beeinflußt werden - ein signifikantes Ziel von Gestaltungsstrategien für den Entwickler und Betreiber von Computersystemen!

Diese Aussagen bezüglich der 3 o.g. Aktivitätenarten des EDV-Benutzers sind gravierend, da sie bestimmen oder zeigen:

- die Ausbildung eines (tatsächlichen und damit realen) Kommunikationsniveaus,
- die Kompliziertheit einer wirksamen Einflußnahme auf die Kommunikationsbeziehungen in der EDV-Nutzung,
- Kommunikation und Steuerung als Prozesse (u.a. Benutzeraktivitäten), die nicht unmittelbar (!) an eine interaktive Rechnernutzung im Dialogmodus gebunden sind, sondern weit über diese zeitliche und inhaltliche Arbeitsphase hinausreichen.

Wird das Kommunikationsniveau D bei einer beliebigen EDV-Nutzung verstanden als Grad der aktivierten Dialogfähigkeit zwischen Benutzer mit seiner EDV-Aufgabe einerseits (Nutzersystem) und dem Computer und dem Betreiber von Computern als EDVsystem andererseits, so ergeben sich folgende Einflußgrössen auf D (Abb. 5 ):

1.Funktionsteilung zwischen Benutzer und dem EDVsystem in bezug auf Kommunikations- und Steuerungsaktivitäten.

$D_N$ = Aktivitäten des Benutzers( Kommunikation und Steuerung)

$D_E$ = Aktivitäten des Betreibers und Computers

Dabei gilt:

$D_N > D_E \rightarrow$ die EDVsystem-Unterstützung tendiert zum Minimum

und damit zu einer hohen Nutzerflexibilität.

$D_N = D_E$ → Maximum an Dialogfähigkeit, das aber sehr kompliziert zu projektieren und zu beherrschen ist.

$D_N < D_E$ → die EDVsystem-Unterstützung tendiert zum Maximum und damit zu einem starren (fixierten) Leistungsangebot mit geringen Nutzereingriffen (z.B. bei Informationssystemen).

Die Relation Flexibilität Starrheit bestimmt in der EDV-Praxis viele soziale Auswirkungen der EDV-Nutzung, vor allem bezüglich des Niveaus von computergestützten Arbeitstätigkeiten, und damit das Niveau von gebliebenen oder verbliebenen Arbeitsinhalten für den Menschen.

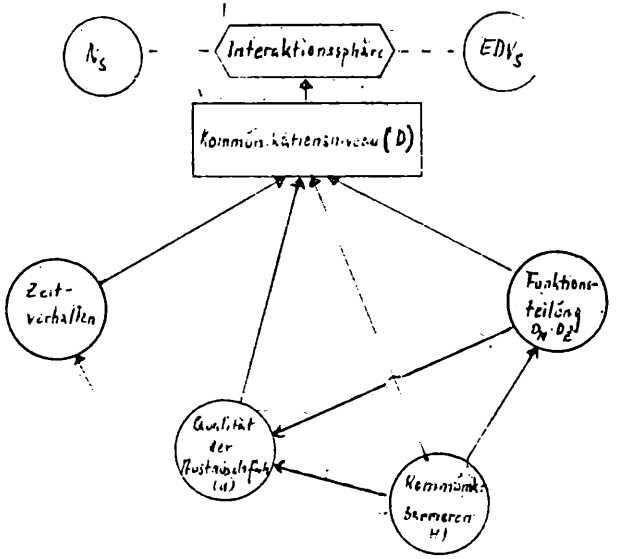



Abb. Modell der Abhängigkeit des Kommunikationsniveaus D
Einflußgrößen

2. Das Zeitverhalten, d.h., die Reaktionszeiten des Partners( -systems) auf Fragen und Aufforderungen des Initiators. Sie bewegen sich für den Partner zwischen einem Maximum und einem Minimum. In diesem Toleranzraum, der individuell schwankt , muß sich ein Optimum befinden, das eine sogenannte " Erträglichkeit der Reaktionszeit für den Partner sichert. Nach MARTIN / 6 / liegt die Toleranzbreite für den Computernutzer zwischen 4 sec und 0,1 sec.

3. Die Qualität der Austauschfähigkeit, d.h., die Adaption kommunikativer Aktivitäten ( Steuerungs- und Kommunikationsaktivitäten) in bezug auf die "Verständnis-" Reaktion (Handlungsreaktion) des Partners. Man kann diese Größe durch einen Koeffizienten d darstellen.
Es gilt dann:

$$0 < d \leq 1 ,$$

wobei bei $D_N = D_E$ der Koeffizient einem Minimum entgegenstrebt. Der Koeffizient d , den man bewerten kann mit Hilfe von Skalierungsverfahren für Kriterien von D , wird also beeinflußt von der Beziehung $D_N$ $D_E$ , da mit wachsender Funktionsteilung die Qualität oder Güte der Dialogfähigkeit aus den o.g. Gründen (Anpassungsschwierigkeiten und -umfang) erschwert erreichbar und beherrschbar wird.

4. Kommunikationsbarrieren (H)
Kommunikationsbarrieren werden verstanden als Differenz zwischen projektierten und tatsächlichen Kommunikationsniveau D . Ursächlich werden die Kommunikationsbarrieren von der Funktionsteilung, der Güte der Austauschfähigkeit und dem Zeitverhalten beeinflußt, d.h., je höher eine der drei Einflußgrössen effektiv an D partizipiert, umso größer wird der Einfluß von H auf diese Einflußgröße rekursiv sein. Diese Ableitung ist nur gegeben, wenn man die Kommunikationsbarrieren als eine Art "Fehler" auffaßt, deren Anzahl oder Aufkommen stochastisch mit der quantitativen Zunahme seiner Einflußgrößen auf D anwachsen dürfte. So wird bei ausgeglichener Relation $D_N$ $D_E$ infolge des notwendigen hohen Automatisierungsaufwandes und Anpassungsaufwandes von Kommunikationsbeziehungen auf informationeller Basis zwangs-

läufig der Anteil und Einfluß von Kommunikationsbarrieren zunehmen.
Alle genannten Einflußgrössen sollten das Objekt von Gestaltungs-strategien des Betreibers und des Entwicklers von Computern in bezug auf EDV-Anwendungen bilden.
Zwischen den o.g. Einflußgrössen von D bestehen teilweise komplizierte Wechselwirkungen und Abhängigkeiten.
Andererseits stellen der Anteil und das Niveau der aufgeführten drei Aktivitätengruppen in der EDV-Nutzung für den Analytiker eine wesentliche Aussage über den Charakter der konkreten EDV-Nutzung in bezug auf kreativitätsfördernde oder persönlichkeitsentleerende Arbeitsinhalte von Tätigkeiten dar. Sie indizieren somit wichtige soziale Auswirkungen der Computerisierung.
Allen genannten Aktivitäten liegt eine Vielzahl von Entscheidungen und damit von Entscheidungsprozessen zugrunde, denen sich der EDV-Benutzer stellen muß, oder die ihm das Rechnersystem abnimmt. Diese Feststellung beinhaltet die eigentliche Ursache für unmittelbare soziale, d.h., auf den EDV-Benutzer bezogene, Auswirkungen der Informationstechnologie im Arbeitsprozeß.

In Anlehnung an die Struktur perzeptiv-intellektueller Leistungen nach PLATH / 17 / lassen sich für die einzelnen Tätigkeitsarten in der EDV-Nutzung unterschiedliche Aussagen anführen, wobei wir nachfolgend 6 Tätigkeitsarten zugrunde legen. Abb. 6 vermittelt eine entsprechende matrixartige Übersicht. Zu den Tätigkeitsarten gehören:

- Dateneingaben , z.B. Patienteninformationen
- Dateneingaben und vordefinierte Ausgaben , z.B. rechnergestütztes Bestrahlungsplanungssystem
- nur vordefinierte Ausgaben , z.B. Fahrplanmitteilungen per Display
- vordefinierte Ausgaben auf Anfragen, z.B. Literaturrechercheсystem
- Instruktionen für DV-Prozesse , z.B. Methodenbank (Benutzung)
- Programmentwicklung (Programmierung).

| Tätigk.-arten i. perzept.-kogn. u. EDV-Nutzg. Leistungen | Daten-eingabe | DE + vordef. Ausg. | nur vordef. Ausg. | Vordef. Ausg. nach Anfrage | Instrukt. f. DV | Programmierung |
|---|---|---|---|---|---|---|
| Orientierungslstg. (Arten aufzunahm., zu erkenn. Signale) | | | | | | |
| Kontrollop. b.d.Vorbertg.d.Ausführg. + b.Ausführg. selbst | | | | | | |
| Entschlüss.+Umsetzen v. Informationen | | | | | | |
| Vorstellgsmäß. Nachvollzieh. vorgegeb. tätgk.-bedeuts.Sachverhalte | | | | | | |
| Beurteilen+Bewerten | | | | | | |
| Kalkul.Auswählen aus mehrer. Mögl., Entscheiden | | | | | | |
| Bereiche zu aktualis. Wissens | | | | | | |
| Entwerfen (selbst. Planung künft. Handl. | | | | | | |
| objekt. Freiheitsgrade f. Arbeitshandlungen | | | | | | |

Abb. 6 Struktur der EDV-Nutzungen (Arten) nach perzeptiv-kognitiven Merkmalen

Erst bei letzterer, d.h., der Programmentwicklung, werden Kenntnisse einer Programmiersprache benötigt. Andererseits setzt sich sich international der Anteil der übrigen ( 5 ) Tätigkeitsarten in der Informationstechnologie gegenüber der Programmentwicklung immer stärker durch. Diese Entwicklung provoziert geradezu Untersuchungen der Rolle und Funktion des EDV-Benutzers in der EDV-Anwendung.

Die EDV-Nutzung sollte also nicht zu eng nur unter dem Aspekt der Programmierung gesehen werden. Die eigentlichen sozialen

Auswirkungen entstehen vordergründig bei den anderen o.g. Tätigkeitsarten der EDV-Nutzung.
Die perzeptiv-kognitiven Leistungen umfassen (nach PLATH):
- Orientierungsleistungen
- Kontrolloperationen
- Entschlüsselung von Informationen
- Vorstellungsmäßiges Nachvollziehen bekannter Sachverhalte
- Beurteilen
- Entscheiden
- zu aktualisierendes Wissen
- Entwerfen
- objektive Freiheitsgrade

Die dargestellte Matrix läßt vermuten, daß sich das Niveau der geforderten perzeptiv-kognitiven Leistungen in Abhängigkeit von den jeweiligen EDV-Tätigkeitsarten verändert und in der Matrix allgemein von links nach rechts zunimmt, bei der Programmierung den Höhepunkt erreicht.

## 5. Die arbeitsmäßige Umgebung des EDV-Benutzers

Als arbeitsmäßige Umgebung des EDV-Benutzers sollten gesehen werden:
- die EDV-Aufgabe und ihre Lösungsanforderungen
- der Umgang mit dem Rechnersystem
- die Integration in eine kooperative Gruppe
- die Arbeitsumgebung mit ihren Anforderungs-, Ziel- und Wertsystemen

Damit existiert auch eine soziale Kommunikation, die wesentlich die Motivation und Einstellung des Benutzers zur EDV stimuliert. Abb. 7 vermittelt diesbezüglich einen Überblick über die Intensität ausgewählter Merkmale von Benutzergruppen am Beispiel der Forschung. Es wurden 6 Benutzergruppen untersucht:
- die Systemprogrammierer, d.h., Mitarbeiter eines Rechenzentrums
- die Anwendungsprogrammierer
- Fachwissenschaftler mit gesamter eigener Programmentwicklung
- Fachwissenschaftler; definieren nur EDV-Aufgabe und nutzen Berechnungsergebnisse

| Merkmale / EDV-Benutz.-gruppen | Anteil an der Progr.-entwicklung | Kooperation bezügl. EDV mit ander. Gruppen | eigene EDV-Qualifik. | Inanspr.d. Nutzerberatung im Rechenzentrum | Niveau d. EDV-Aufgabenspezifikation | Niveau rechentechnischer Anforderungen |
|---|---|---|---|---|---|---|
| Systemprogrammierer | sehr hoch | sehr gering | sehr hoch | - | sehr hoch | sehr hoch |
| Anwendungsprogrammierer | hoch | hoch | sehr hoch | sehr gering | (hoch) | sehr hoch/ hoch |
| Fachwissensch.; gesamte Progr.-entwicklung | sehr hoch | gering | sehr hoch/ hoch | hoch/ gering | sehr hoch | hoch |
| Fachwiss.; def. Aufgabe + Auswertungv. Erg. | gering | sehr hoch | mittel/ gering | sehr hoch | mittel | mittel/ gering |
| Fachwiss.; EDVA-Abarbeitg.+ Auswertg. | sehr gering | gering | gering | gering | (gering) | hoch/ gering |
| übr. Fachwiss. | - | sehr gering/ - | sehr gering | sehr gering/ - | - | - |

Abb. 7 Intensität der Merkmale von EDV-Benutzergruppen in einem Forschungsbereich

- übrige Fachwissenschaftler; nutzen nicht die EDV

Die untersuchten Merkmale zeigen ein unterschiedliches Bild der sozialen Kommunikation auf dem Gebiet der EDV: In Abhängigkeit vom persönlichen Arbeitsanteil an der EDV-Nutzung verändert sich ihr Niveau. Es zeigt sich , daß die zwischenmenschliche Kommunikation durchaus als wichtige Ergänzung und nicht als Alternative zur Mensch-Rechner-Kommunikation zu bewerten ist, wenn der EDV-Benutzer als Mittelpunkt und Träger der EDV-Nutzung gesehen wird.

Wird der Benutzer bedeutungs- oder rollenmäßig in den Mittelpunkt von EDV-Nutzungen bezüglich ihrer Projektierung und Gestaltung gestellt und letzlich die EDV-Anwendung mit ihren Arbeitsergebnissen als"Finalprodukt" gesehen, so müssen zusätzlich auch kognitive und soziale Aspekte beim EDV-Einsatz berücksichtigt werden.

Unter diesen Bedingungen läßt sich ein"Benutzerbild" als Zustand eines Menschen in der EDV-Nutzung beschreiben, der sich über ein EDV-Nutzungsverhalten individuell äußert, aktiviert und damit zu einer leistungsbeeinflussenden Komponente im Mensch-Maschine-System wird. Unter dieser Voraussetzung kann der EDV-Benutzer nur in Verbindung mit seiner Umgebung, d.h., in seiner gesellschaftlichen Determiniertheit, bewertet werden. Damit abstrahieren wir bewußt von einer Aufzählung charakterologischer Merkmale als Benutzerbild-Interpretation, da diese individuell zu verschieden sind und nicht die wechselseitigen Beziehungen zur Umwelt ausreichend und verallgemeinerungsfähig reflektieren.

## 6. Verhalten, Kommunikation und Organisation als zentrale Begriffe in der EDV-Nutzung

Mit diesen Aussagen wird demonstriert, daß Verhalten, Kommunikation und Organisation inhaltlich sehr eng mit der Beschreibung des EDV-Benutzers verbunden sind und künftig einen hohen Stellenwert in der Informatik erhalten sollten. Eine Beeinflussung des Benutzers wird nur über die inhaltliche Berücksichtigung dieser Begriffe bei der Ausbildung der EDV-Nutzung als oder im Arbeitsprozeß möglich. So wird schon jetzt

erkennbar, daß Inhalt und Bereitstellung hocheffizienter Kommunikationsbeziehungen, oft nur für kurze Zeiträume, z.B. während einer Terminalsitzung, ein signifikantes Gestaltungsziel für EDV-Nutzungen bilden, die letztendlich auch die Einstellung des Benutzers zur EDV und damit auch die Ausnutzung der tatsächlichen Möglichkeiten des Computers beeinflussen.
Nachfolgend soll überblicksartig das Modell der Einflußnahme des EDV-Benutzers und seiner Position in der EDV vorgestellt werden, das insbesondere die Bedeutung und wachsende Brisanz der o.g. 3 Begriffe betont (Abb. 8 ). Dabei werden die arbeitsmäßigen und psychologischen Einflußnahmen von 6 Komponenten aufgezeigt und zwar der Komponenten:

- fachliches Problem
- abgeleitete EDV-Aufgabe
- EDV-Nutzung als Arbeitstätigkeit
- Computer
- Effekt der EDV-Leistung bzw. des Finalprodukts
- kooperierende Gruppe

Es wird das Verhalten des Benutzers als Input- bzw. Outputgröße aufgefaßt, die sich innerhalb des Bezugrahmens oder Kontextes Kommunikation und Organisation modifiziert, wobei diesen Funktionen ein Wechsel von kollektiver und individueller Tätigkeit als typisches Merkmal vor allem der Programmentwicklung inhärent ist. Interessant bei dieser Modellvorstellung ist der Einfluß der kooperierenden Gruppe im Vorfeld der EDV-Nutzung auf den EDV-Benutzer, d.h., auf seine Persönlichkeit und das Verhalten als aktivierte Persönlichkeit.
Zusammenfassend können wir feststellen und damit als Definition fixieren:
Unter einem EDV-Benutzer soll verstanden werden ein Individuum mit einer EDV-Aufgabe, das zu ihrer Lösung gezielt EDV-Leistungen im Rahmen einer Mensch-Rechner-Kombination, insbesonders Leistungen des Rechners in Anspruch nimmt. Dazu sind Leistungen des Benutzers, teilweise in Kooperation, zwecks Verarbeitung und Aufbereitung dieser Inanspruchnahme erforderlich. Es entstehen bei der EDV-Nutzung als Arbeitstätigkeit

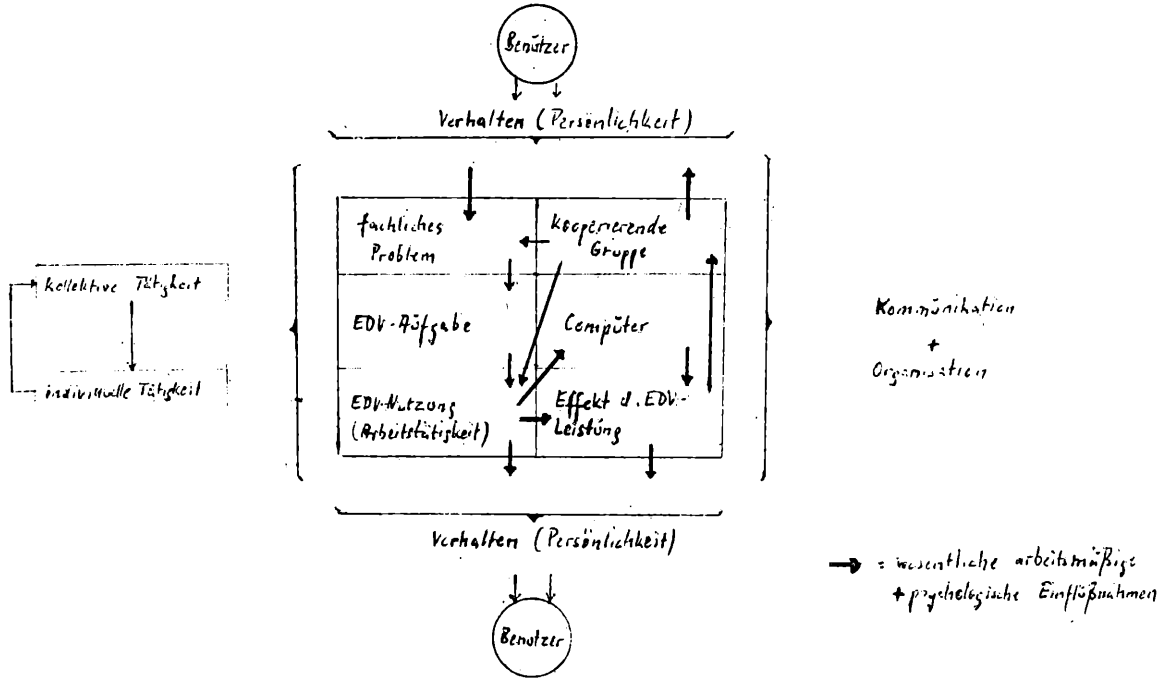



Abb. 8 Modell der Einflußnahme des EDV-Benutzers und seiner Position in der EDV (vereinfacht)

spezielle Verhaltensweisen und Kommunikationsbeziehungen, die in Wechselwirkung mit dem Rechner und/oder Rechnerbetreiber (Rechenzentrum) notwendig sind, und vom EDV-Benutzer mit einem vorwiegend perzeptiv-kognitiven Tätigkeitsaufkommen realisiert werden. Das Niveau der arbeitsmäßigen Funktions- und Aufgabenteilung Benutzer - Rechner beeinflußt soziale Auswirkungen.

Aus dem Obengesagten lassen sich für die Projektierung von EDV-Nutzungen einige interessante Schlußfolgerungen ableiten:

1. Das Verhalten sollte als wesentliche Komponente für die Organisation der EDV-Nutzung als Arbeitstätigkeit in bezug auf die soziale Integration, Gestaltung der Aufgabenteilung Mensch - Rechner, Ausbildung ihrer Kommunikationsbeziehungen sowie der Entwicklung des rechentechnischen Einsatzes unter dem Primat der Nutzungsfreundlichkeit berücksichtigt werden.
2. Die Kommunikation ist als weitere wesentliche Komponente in bezug auf die Funktionsfähigkeit des Computereinsatzes, Realisierung arbeitswissenschaftlicher Aspekte bei der EDV-Nutzung sowie zur Produktivitäts- und Kreativitätserhöhung für den EDV-Benutzer einzubeziehen.
3. Auffassung und Gestaltung des EDV-Einsatzes als gesellschaftliches Anliegen, das nur interdisziplinär und unter Zugrundelegung von Leistungs-, Beanspruchungs- und persönlichkeitsbildenden Faktoren als Einheit bei gleichzeitiger Berücksichtigung der sozialen und ökonomischen Effizienz und Auswirkungen optimal zu realisieren ist.
4. Die Auffassung und Behandlung des EDV-Benutzers als "Durchschnitts- oder typischen Nutzer" mit typischen Anforderungen und Verhaltensweisen ist nicht mehr ausreichend und führt über die übliche Einordnung in Nutzerkategorien nur zu bedingtem Aussagewert. Diese Aussage ist wichtig in Verbindung mit der Nachnutzungsproblematik in der EDV.
5. Es besteht die Möglichkeit und Notwendigkeit einer Einflußnahme auf die EDV-Nutzung und damit auf den Benutzer durch Entwickler und Betreiber von Rechnern über die Ausbildung von geeigneten Kommunikationsbeziehungen. Diese Einfluß-

nahme ist aber begrenzt aus fachlichen, zeitlichen und sozialpsychologischen, auch lokalen Gründen und sollte sich beziehen auf die unmittelbare Unterstützung und Stimulierung der EDV-Nutzung als Arbeitstätigkeit.

## 7. Schlußfolgerungen

Zusammengefaßt ergeben sich verschiedene neue Fragestellungen für die Informatik in Verbindung mit der Erforschung des EDV-Benutzers wie:

- die Erforschung der Motivation zur EDV-Anwendung
- Eignungsgrenzen des Menschen in der EDV-Nutzung und ihre Berücksichtigung bzw. evtl. Erweiterung bei der bzw. mit der Hilfe der Gestaltung von rechnergestützten Informationssystemen und anderen Nutzungsarten der EDV
- Inhalt und Gestaltung persönlichkeitsbildender Faktoren bei der EDV-Nutzung
- Kommunikationsgestaltung als Organisationsproblem in der EDV-Nutzung
- individuelle Leistungs- und Persönlichkeitsunterschiede und ihre Berücksichtigung beim Einsatz dezentralisierter Informationstechnologien
- die Berücksichtigung bzw. Adaption von computerseitigen Innovationen an das EDV-Benutzerverhalten und seine Merkmale
- Normierung und individuelle Freizügigkeit bei der Programmentwicklung und überhaupt in der EDV-Nutzung - nicht als Widerspruch sondern als wechselseitige Ergänzung!
- die EDV-Nutzungsfreundlichkeit unter Berücksichtigung kognitiver und sozialer Aspekte
- das EDV-Nutzungsverhalten bei unterschiedlichen Informationstechnologien
- die Rolle und Auswirkungen der EDV im fachbezogenen Arbeitsprozeß des EDV-Benutzers (u.a. Motivations- und Einstellungshintergründe)
- Auswirkungen von EDV-Nutzungen auf die Persönlichkeit des Benutzers.

Diese beispielartig aufgeführten Fragestellungen kreieren neben der Hardware und Software auch den Benutzer, den Men-

schen, als Forschungsobjekt in der Informatik.
Dabei sollten sich 4 Forschungsschwerpunkte herausbilden und zentrieren:

- das Nutzerverhalten
  - als psychologisch-soziales Phänomen unter den Bedingungen der EDV-Nutzung
- die EDV-Nutzung als Arbeitstätigkeit
  - vorrangig als arbeitswissenschaftliches Phänomen unter Berücksichtigung sozialer und ökonomischer Wirksamkeitsuntersuchungen des konkreten EDV-Einsatzes
- die Kommunikationsgestaltung bei EDV-Nutzungen
  - vorrangig als Organisationsphänomen mit dem Ziel der Schaffung von Entfaltungsbedingungen für Nutzersysteme
- gesellschaftliche und soziale Auswirkungen von Informationstechnologien
  - im Sinne von Fern- und Nachwirkungen, wie z.B. Datenschutz, juristische Fragen

Vorliegende Ausführungen sollen Anregungen in diesem Sinne vermitteln.

Literaturhinweise


/1/ Sackman,H. Man Computer Problem Solving. Experimental Evualation of Time Sharing and Batch Processing
Princeton NNew York: Auerbach Publishers Inc. 1970

/2/ Newell,A., Simon,H.A. Human Problem Solving
Prentice Hall,N.Y.: Englewood Cliffs 1972

/3/ Nickerson,R.S. Man-Computer Interaction: A Challenge for Human Factors Research
Ergonomics 12(1969)4, 501-517

/4/ Weizenbaum,J. Computer Power and Human Reason
San Francisco: Freeman 1976

/5/ Bjørn-Andørsen, N., Borum,F. Demokratisierung der Gestaltung von Informationssystemen
in: Mensch und Computer. Zur Kontroverse über die ökonomischen und gesellschaftlichen Auswirkungen der EDV (Hrsg.: H.R.Hansen, K.T.Schröder, H.J. Weihe)
München, Wien: R. Oldenbourg 1979, 327 - 347

/6/ Martin,J. Design of Man-Machine Dialogues
Prentice Hall, N.Y.: Englewood Cliffs 1975

/7/ Fischer,U.E. Teilnehmerbetrieb
Köln-Braunsfeld: Rudolf Müller 1973

/8/ Mumford,E., Sackman,H.(eds.) Human Choice and Computers
Amsterdam: North Holland Comp. 1975

/9/ Hansen,R., Schröder,K.T., Weihe,H.J. (eds.) Mensch und Computer. Zur Kontroverse über die ökonomischen und gesellschaftlichen Auswirkungen der EDV
München, Wien: R.Oldenbourg 1979

/10/ Weinberg, G.M. The Psychology of Computer Programming
New York: Van Nostrand Reinhold 1971

/11/ Shneiderman,B. Software Psychology. Human Factors in Computer and Information Systems Cambridge, Mass.: Winthrop Publishers Inc. 1980

/12/ Boies,S.J. User behavior in an interactive computer system IBM Systems Journal 11(1974)13,1-18

/13/ Mc Cormick,E.J. Human Factors Engineering New York: Mc Graw-Hill 1970

/14/ Neumann,J., Timpe,K.-P. Psychologische Arbeitsgestaltung Berlin: Deutscher Verlag der Wissenschaften 1976

/15/ Pilgrim,J. Zur Benutzer-EDV-Kommunikation im Bereich der Forschung Konferenz Strategienbildung in Wissenschaft und Technik, Berlin 1978 in: Wissenschaftswissenschaftliche Beiträge Heft 8, Humboldt-Universität Berlin 1979, 128-136

/16/ Eason,K.D. Dialogue design implications of task allocations between man and computer Ergonomics 23(1980)9, 881-891

/17/ Plath,H.-E. Tätigkeitsanforderungen und Beanspruchungsfolgen bei Arbeitsmitteln unterschiedlicher Technisierungsstufen und verschiedenen Graden innerbetrieblicher Arbeitsteilung in:Optimierung von kognitiven Arbeitsanforderungen (Hrsg.:W.Hacker und H.Raum) Berlin: Deutscher Verlag der Wissenschaften 1980, 144-162

Verfasser:

Dr. sc. Jürgen Pilgrim
ZI für Molekularbiologie der AdW der DDR
Rechenzentrum
1115 Berlin
Lindenberger Weg 70

Herrlich, O., Detering, R.

# Anforderungen an Spezifikationsmittel im Programmentwicklungsprozeß

## 1. Einleitung

Im Verlauf der stürmischen Entwicklung auf dem Gebiet der Software- Technologie haben sich Inhalt und Breite des Begriffs Spezifikation verändert. Neben der stärkeren Betonung früher Entwurfsstadien werden Spezifikationsmittel auch in der Phase der Installation/Wartung eingesetzt, sodaß ihre Nutzung praktisch alle Phasen des Lebenszyklus von Software umfaßt.
Die Erweiterung des Aufgabenspektrums ist in engem Zusammenhang mit den erweiterten Auffassungen des Software- Lebenszyklus zu sehen, da sich die Aufstellung der Spezifikationen und ihre Anwendung im Gesamtprozess der Software- Produktion einordnen, siehe /FLKO/, /COCO/, /HÜNK/, /RIFA/, /HOFF/.
Trotz dieser Ausweitung des Einsatzspektrums bleiben Spezifikationen im wesentlichen Aufgabenbeschreibungen, die auf deskriptive Weise ausdrücken, was gefordert ist, im Gegensatz zum wie der programmiersprachlichen Formulierung.

## 2. Einflussgrößen für Anfoderungen

Ein wesentlicher Gesichtspunkt im Rahmen der Software- Technologie ist die Erstellung von Software mit Produktcharakter.
Zur näheren Erläuterung des Begriffs können verschiedene Punkte angeführt werden.

- Entwickler und Nutzer sind nicht identisch.
- Es arbeitet eine größere Gruppe von Entwicklern.
- Die Software ist Bestandteil anderer Produkte.
- Es sind mehrere Varianten der Software vorgesehen.

Für sogeartete Software- Produkte erlangt die Frage der zugehörigen Spezifikation über den gesamte Prozess ihrer Erstellung verstärkte Bedeutung, da sie Qualität, Nutzbarkeit, Lebensdauer usw. beeinflussen.
Art, Umfang und Anforderungen an die Spezifikationen hängen dabei vom Umfeld der zu lösenden Probleme sowie deren Randbedingungen ab. Neben der ökonomischen Zielstellung, die Forderungen hinsichtlich Nachnutzbarkeit, Portabilität und Lebensdauer der Software impliziert, ist das Einsatzgebiet von

großer Bedeutung. So ist für die Spezifikation sicherheitsrelevanter Systeme wie Verkehrssteuerungen ein höherer Aufwand bzgl. Genauigkeit und Überprüfbarkeit notwendig als für kommerzielle probleme Probleme der ökonomischen Datenverarbeitung, z.B./GOTT/,/VOGE/. Außerdem spielen die Qualifikation und Vorstellungswelt des Nutzers eine Rolle, um für diesen die Spezifikation während der Software- Entwicklung und Nutzung sowie bei der Beschreibung des Ausgangsproblems verständlich zu machen.Last not least entstehen im Produktionsprozess der Software Anforderungen an die Spezifikation, deren Wichtung für die einzelnen Entwicklungsphasen unterschiedlich ist. Die Anforderungen, die durch die oben genannten Faktoren beeinflußt werden, sollen im folgenden anhand der Entwicklungsphasen der Software untersucht werden. In diesem Zusammenhang setzen sich international verstärkt Auffassungen einer dualen Sicht der Software-Produktion durch, die neben der Dekomposition in kleinere Teile beim Systementwurf dual dazu der Zusammenbau der Einheiten bei der Systemintegration betrachten. Im Zusammenhang mit Spezifikationsaktivitäten ergibt sich ähnliches, da eine Dualität zwischen der Beschreibung der Funktion der Bausteine (Spezifikation) und der Überprüfung der Funktion der Bausteine (Validation) besteht.

Demzufolge werden nicht nur Anforderungen zum Beschreibungsaspekt, sondern auch zum Überprüfungsaspekt betrachtet.

- Spezifikationen müssen möglichst abstrakt sein, d.h. noch von der programmtechnischen Realisierung entfernt sein.
- Sie müssen schon so formal sein, daß sie automatisiert verarbeitet werden können.
- Sie müssen es erlauben, Unvollständigkeiten auszudrücken.
- Sie müssen flexibel handhabbar, d.h. leicht änderbar sein.
- Sie müssen für alle am Entwurf Beteiligten verständlich sein.
- Sie müssen es erlauben, das gewünschte Systemverhalten korrekt auszudrücken.
- Sie müssen die Überprüfung ihrer Gültigkeit erlauben, d.h.
    - der Vollständigkeit,
    - der Widerspruchsfreiheit und
    - der Konsistenz.

Diese Anforderungen sind für unterschiedliche Anwendungen und Entwicklungsphasen der Software von unterschiedlicher Wichtung. Das schlägt sich auch in verschiedenen Systematisierungsversuchen in der Literatur nieder, worin Spezifikationsmittel und -methoden nach bestimmten Anfoderungen bzw. Einsatzgebieten geordnet werden. So wird bei /HESS/ in der von ihm 'Softwareentwicklungslandschaft' genannten Darstellung nach Formalität, Abstraktheit und Automatisierbarkeit geordnet.
Im folgenden soll eine Systematisierung der Anforderungen anhand ihrer Wichtung im Verlauf des Entwicklungsprozesses versucht werden. Die Maßstäbe sind symbolischer Natur, da sich nicht auf ein konkretes Phasenmodell der Software- Entwicklung beschränkt werden soll; der Variantenreichtum für diese Modelle ist außerordentlich groß.

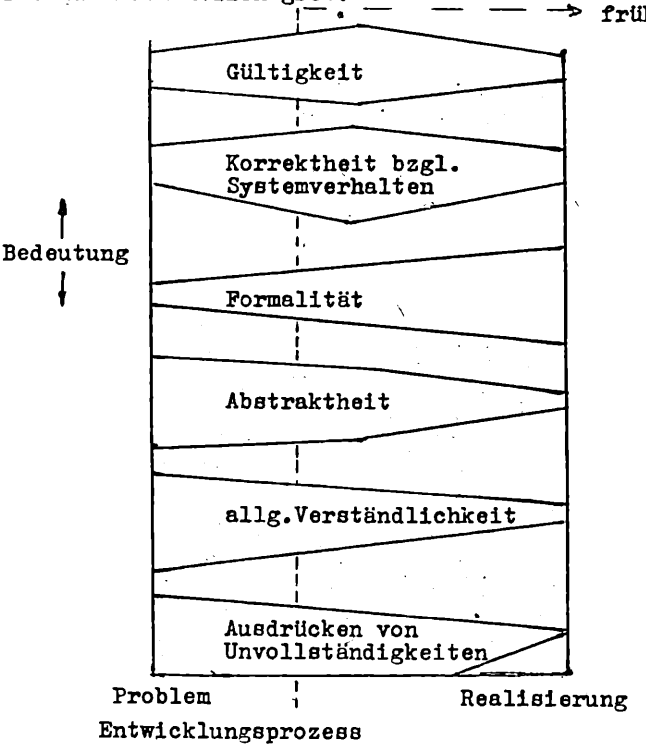



Abb. 1: Wichtung der Anforderungen an Spezifikationsmittel

Um Schwerpunkte für die Software- Produktion zu setzen, müssen die Einflüsse der Entwurfsphasen auf die Qualität der Software untersucht werden. Betrachtet man z.B. die Fehlerquote bzgl. der Gesamtfehler für die frühe Entwurfsphase, so beträgt diese ca. 50%, wobei aber dafür 80% der gesamten Fehlerbeseitigungskosten anfallen /BEICH/. Diese hohen Kosten entstehen dadurch, daß spät gefundene Entwurfsfehler aufwendige Korrekturen erfordern. Bei der Betrachtung des Gesamtzeitaufwandes werden in der Literatur durchschnittlich 50% der frühen Entwurfsphase zugeordnet. In dieser Phase sind viele grundsätzliche Probleme zu lösen und wichtige Entscheidungen für die nächsten Entwurfsschritte bis hin zur Realisierung zu treffen. Daher erscheint eine Konzentration auf Probleme der frühen Entwurfsetappen dringend geboten.

3. Probleme der frühen Entwurfsphase

Bei der Betrachtung dieser Phase erweist sich die Darstellung der Anforderungs- Wichtung als nützlich.
Es gilt für die zu lösenden Software- Probleme eine Darstellung von hoher Verständlichkeit zu finden, die aber trotzdem auf Grund ihrer wachsenden Formalität schon Überprüfungen zum gewünschten Systemverhalten und zur Gültigkeit zulässt.
Die Formulierung der Probleme besitzt noch einen hohen Grad an Abstraktheit, d.h. sie ist noch sehr maschinenfern, und erlaubt solche Unvollständigkeiten auszudrücken, die durch die noch nicht abgeschlossene Problemanalyse verursacht werden.
In dieser Phase wird angestrebt, das Ergebnis der Problemanalyse möglichst informal auszudrücken, aber schon Möglichkeiten für die rechnergestützte Verarbeitung und Überprüfung zu schaffen. Es wird somit ein Modell der Problemlösung aufgestellt, das eine anschauliche, fachübergreifende (bzgl. des Bearbeiterkreises) Darstellung erlaubt und für alle am Entwurf Beteiligten die Entwicklungsschritte verständlich und kritisierbar machen soll.
In Anlehnung an die Anforderungen an Spezifikationsmittel können abgeleitete Forderungen an ein Modell in einer frühen Entwurfsetappe formuliert werden.

Ein Modell
- muß die wesentlichen Eigenschaften des Objekts beschreiben.
  (Gültigkeit bzgl. Systemverhalten)
- muß 'in irgend einem Sinne leichter zugänglich sein'.
  (Verständlichkeit)
- muß flexibel und änderbar sein.
  (informal, Ausdruck von Unvollständigkeiten)
- darf nur Funktionen des zu realisierenden Systems beschreiben, die durch dessen innere Logik bestimmt sind.
  (Abstraktheit)
- muß analysierbar sein, sowohl manuell als auch automatisiert.
  (Formalität, Verständlichkeit, Gültigkeit)

In einem solchen Modell werden alle Anforderungen an das zu realisierende System zusammengefaßt, zur Bezeichnung dieser Anforderungen hat sich in der Literatur der englische Begriff requrement eingebürgert. Dieser soll auch hier verwendet werden, um Verwechselungen zu dem im Titel genannten Begriff zu vermeiden.

Zur Aufstellung und Bearbeitung der requirements sind vielfältige Aktivitäten notwendig, siehe /KSCH /,/HAMÜ /, die im nachfolgenden Bild zusammengefaßt werden sollen.

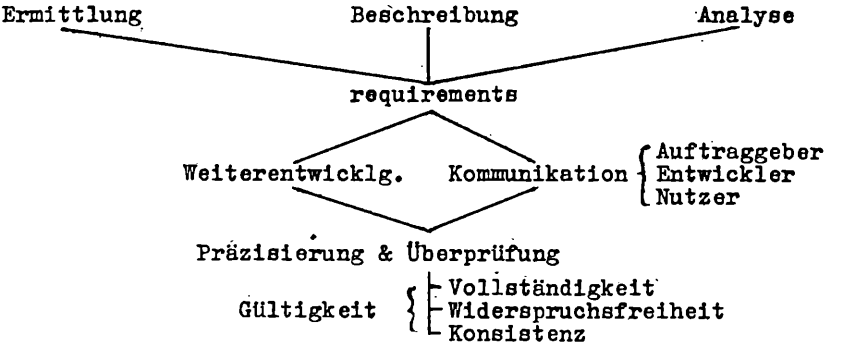



Abb. 2: requirement- Aktivitäten

Neben zahlreichen anderen Problemen sollen hier abschließend zwei Hauptprobleme der requirement- Aktivitäten angedeutet werden.

1. Die Aktivitäten zur requirement- Spezifikation stellen keinen deterministischen Prozess dar. Auf Grund der enthaltenen Unvollständigkeiten, der hohen Flexibilität sowie der Vielschichtigkeit des Entwurfsprozesses sind der Automatisierung Grenzen gesetzt. Daher wird auf diesem Gebiet eine interaktive Arbeitsweise unabdingbar.
2. Besonders in frühen Stadien der Software- Produktion sind fachlich unterschiedlich gelagerte Interessen- und Arbeitsgruppen am Entwurf beteiligt. Grob gesagt handelt es sich dabei um
   - Auftraggeber
   - Entwickler
   - Nutzer.

   Bei der dabei nötigen Kommunikation zwischen den Gruppen ist eine interdisziplinäre Arbeitsweise unbedingt notwendig. Grob verallgemeinert kann man den Ablauf folgendermaßen darstellen.

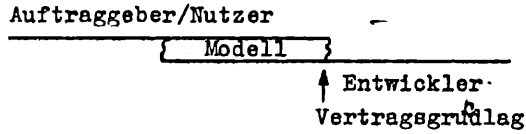



Abb. 3: Interdisziplinäres Entwurfsschema

Gerade für die Realisierung von Software mit Produktcharakter ist dieser überlappenden Etappe große Aufmerksamkeit zu schenken, um eine wohldefinierte Arbeitsgrundlage zu erhalten.

4. Literatur

/BEICH/ Beichter, F.
Spezifikationssprachen in der industriellen Praxis
in /FLKO/, S. 300-302

/COCO/ Cotterman, W.W., Couger, J.D., Enger, N.L., Harold, F. (ed.)
Systems Analysis and Design
Proc. Conf. and Workshop, Georgia State University,
Atlanta 8.-10.9.1980
North Holland, New York, Amsterdam, Oxford, 1981

/FLKO/ Floyd, C., Kopetz, H. (Hrsg.)
Software Engineering - Entwurf und Spezifikation
Proc. Tagung II/1980 mit Workshop
B.G.Teubner Verlag, Stuttgart, 1981

/GOOS/ Goos, G. (Hrsg.)
Werkzeuge der Programmiertechnik
Proc. GI- Arbeitstagung, Karlsruhe 1981
Informatik- Fachberichte 43
Springer- Verlag, Berlin, Heidelberg, New York, 1981

/GOTT/ Gottschalk, W.
Entwurfshilfsmittel für die Prozessrechnerprogrammierung
Signal+Draht, 70(1978)9, S. 194-198

/HAMÜ/ Hausen, H.L., Müllerburg, M.
Software- Produktionsumgebungen: Entwicklungsstand und Trends
in /GOOS/, S. 1-27

/HESS/ Hesse, W.
Methoden und Werkzeuge zur Software- Entwicklung: Einordnung und Überblick
in /GOOS/, S. 113-153

/HOFF/ Hoffmann, H.-J. (Hrsg.)
Programmiersprachen und Programmentwicklung
Proc. 6.Fachtagung, Darmstadt 1980
Informatik- Fachberichte 25
Springer- Verlag, Berlin, Heidelberg, New York, 1980

/HÜNK/ Hünke, H. (Hrsg.)
Software Engineering Environments
Proc. Symposium Lahnstein 1980
North Holland, New York, Amsterdam, Oxford, 1981

/KSCH/ Krämer, B., Schmidt, H.W.
Interaktive Software- Entwicklung durch schrittweise Formalisierung
in /FLKO/, S. 310-312

/RIFA/ Riddle, W.E., Fairley, R.E. (ed.)
Software Development Tools
Proc. Workshop on Software Development Tools,
Pingree Park, Colorado 1979
Springer- Verlag, Berlin, Heidelberg, New York, 1979

/VOGE/ Voges, U.
Anforderungsspezifikation für sicherheitsrelevante Systeme
in /FLKO/, S. 279-281

Prof.Dr.-Ing.habil. O.Herrlich
Dr.-Ing. R.Detering

TU Dresden
Sektion Informationsverarbeitung
8027 Dresden
Mommsenstraße 13

# Leistungsbewertung im Rechenbetrieb

Doz. Dr. sc. techn. Gerhard Bergholz
Technische Universität Dresden

## 1. Einführung

Mit der gewachsenen Verbreitung der Rechentechnik wurde die effektive Nutzung der Ressourcen einer Rechenanlage zu einem Erfordernis. Der gegenwärtige internationale und nationale Stand ist dadurch gekennzeichnet, daß eine Diskrepanz zwischen den bereitgestellten Ressourcen und ihrer Nutzung vorhanden ist. In der Tabelle 1 werden die prozentualen Auslastungen der Ressourcen: Hauptspeicher (HS), Magnetplattengeräte (MPG), Magnetbandgeräte (MBG) und Paralleldrucker (PD) für eine Rechenanlage EC 1040 angegeben, die aus Messungen gewonnen wurden. Keine der betrachteten Ressourcen ist über 50% ausgelastet.

Zur Beseitigung der oben genannten Diskrepanz wurden im letzten Jahrzehnt international Mittel und Methoden entwickelt, die in der Forschungsdisziplin Leistungsbewertung von Rechnersystemen zusammengefaßt werden. Hauptanwendungsgebiet der Leistungsbewertung von Rechnersystemen ist der praktische Rechenbetrieb. Damit bildet die Leistungsbewertung im Rechenbetrieb eine wichtige wissenschaftliche Grundlage für die Technologie des Rechenbetriebs.

Hauptgebiete der internationalen Forschung auf dem Gebiet der Leistungsbewertung sind: die Schaffung von Verhaltensmodellen (empirische, analytische und Simulationsmodelle) als Grundlage für die Leistungsvorhersage, die Schaffung von Mitteln der Leistungsmessung und schließlich die Nutzung der Verhaltensmodelle und Mittel der Leistungsmessung für die Leistungsverbesserung und den Leistungsvergleich.

Die Ergebnisse der internationalen Forschung werden in Literaturübersichten /1/ und /2/ und einer Anzahl von Büchern, insbesondere in /3/ zusammenfassend dargestellt. Über das Gebiet der Leistungsbewertung von Rechnersystemen fand eine größere Zahl von Spezialtagungen statt, deren Vorträge in Sammelbänden

veröffentlicht werden (siehe z.B. /4/). Seit 1981 erscheint die internationale Zeitschrift "Performance Evaluation".

In diesem Beitrag werden die Begriffe der Leistungsbewertung unter Nutzung konkreter eigener Untersuchungen des Autors dargelegt.

## 2. Leistungsbewertung von Rechnersystemen

Die Leistungsbewertung von Rechnersystemen beschränkt sich auf die Bewertung von Eigenschaften der Rechnerinstallation, die quantitativ bewertbar sind. Der Vorteil dieser Beschränkung besteht darin, daß auf dieser Grundlage eine Theorie aufgebaut werden kann, in der quantitative Größen als wesentliche Parameter eingeschlossen sind.
In der Literatur (siehe z.B. /2/) wird eine größere Zahl von Größen für die Bewertung der Rechnerleistung genutzt. Man kann diese Bewertungsgrößen in drei Hauptklassen einteilen: in Bewertungsgrößen, die das Durchsatzverhalten ausdrücken, in Bewertungsgrößen, die das Reaktionsverhalten ausdrücken und in Bewertungsgrößen, die das Auslastungsverhalten ausdrücken. Konkrete Bewertungsgrößen werden in diesem Beitrag später im Zusammenhang mit konkreten Untersuchungen eingeführt. Der Gegenstand der Leistungsbewertung besteht in der Schaffung von Mitteln und Methoden der Leistungsbewertung und in der Nutzung dieser Mittel und Methoden. Leistungsbewertungsmittel sind vor allem Verhaltensmodelle und Meßmittel. Nutzungsrichtungen der Leistungsbewertung sind die Leistungsverbesserung, der Leistungsvergleich und die Leistungsvorhersage, z.B. für die Planung.

### 2.1. Verhaltensmodelle

Ein Verhaltensmodell beschreibt den Zusammenhang zwischen unabhängigen Einflußgrößen und Bewertungsgrößen B. Die unabhängigen Einflußgrößen können in Arbeitslastparameter A und Konfigurationsparameter S unterteilt werden. Formelmäßig kann dann allgemein geschrieben werden

$$B = B(A, S) \qquad (1)$$

und als Blockschema für das Rechnersystem ergibt sich die Abbildung 1. Wir können die Verhaltensmodelle in die drei Modellklassen: empirische Modelle, Simulationsmodelle und analytische Modelle unterteilen. In den Verhaltensmodellen werden Methoden der Wahrscheinlichkeitstheorie und mathematischen Statistik angewendet.
Eine wichtige Eigenschaft eines Verhaltensmodells ist seine Adäquatheit, die man praktisch als Übereinstimmung der mit dem Verhaltensmodell gewonnenen theoretischen Ergebnisse mit den realen Bedingungen am Rechnersystem auffassen kann. Wenn die Anwendung von Verhaltensmodellen zu richtigen Ergebnissen führen soll, dann muß mit adäquaten Verhaltensmodellen untersucht werden. Deshalb bildet die Untersuchung der Adäquatheit von Verhaltensmodellen, die auch Validierung genannt wird, eine wichtige Etappe bei der Aufstellung von Verhaltensmodellen.
Eine Möglichkeit der Untersuchung der Adäquatheit eines Verhaltensmodells besteht in folgendem Vorgehen. Dazu betrachten wir das Rechnersystem mit den Arbeitslastparametern A, den Konfigurationsparametern S und den Bewertungsgrößen B und das Verhaltensmodell des Rechnersystems mit den Arbeitslastparametern A', den Konfigurationsparametern S' und den Bewertungsgrößen B' (siehe Abb. 2) unter den Voraussetzungen

$$\left.\begin{aligned} A &= A' \\ S &= S' \end{aligned}\right\} \qquad (2)$$

Dabei werden die Bewertungsgrößen B und B' verglichen. Zum Beispiel kann als Maß für die Adäquatheit $|B-B'|$ genutzt werden und auf dieser Grundlage der Grad der Übereinstimmung des Verhaltensmodells mit dem realen Rechnersystem bestimmt werden. Die Größen A, S und B des Rechnersystems müssen dabei durch Messungen gewonnen werden.
Wir betrachten jetzt als Beispiel eines Verhaltensmodells die Abhängigkeit der Rechnerleistung L von der Verteilung der Systemdateien über die Plattenstapel und Zylinder der Plattenstapel. Dieser Zusammenhang kann über zwei Modellstufen, Modell der Systemplatten und Leistungsmodell (siehe Abb. 3), hergestellt werden, wobei

$$L = L(\bar{T}_{ps}, M) \qquad (3)$$

und

$$\bar{T}_{ps} = \bar{T}_{ps}(\underline{d}, \underline{\varepsilon}) \qquad (4)$$

gilt. Hierbei werden die Bezeichnungen

L - Rechnerleistung (Durchsatzrate in Arbeitseinheiten/Zeiteinheit, z. B. Anzahl der phys. Sätze/s)

$\bar{T}_{ps}$ - mittlere Positionierzeit aller Systemplatten

M - Multiprogrammfaktor

D1, ..., Dn - Systemdateien

$\underline{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ - Vektor der Plattenstapelnummern der Systemdateien $D_i$ (i=1,...,n)

$\underline{d} = (d_1, \ldots, d_n)$ - Vektor der Zylinderanfangsadressen der Systemdateien $D_i$ (i=1,...,n)

genutzt.

## 2.2. Leistungsmessung

Unter der Bezeichnung Leistungsmessung wird die meßtechnische Erfassung (einschließlich statistische Verdichtung) der Arbeitslastparameter A und der Bewertungsgrößen B verstanden. Die Leistungsmessung bildet gemeinsam mit den Verhaltensmodellen die Grundlage für die Leistungsverbesserung im Rechenbetrieb und für den Leistungsvergleich verschiedener Rechnerkonfigurationen. Sie wird außerdem für die Adäquatheitsüberprüfung der Verhaltensmodelle genutzt.
Gewöhnlich erfolgt die Leistungsmessung in zwei Stufen, in der Stufe der Meßwerterfassung und der Stufe der Meßwertauswertung. Dabei wird bei ereignisorientierter Messung während der Meßwerterfassung eine Ereignisfolge (Trace) erfaßt und aufgezeichnet und während der Meßwertauswertung die den einzelnen Ereignissen zugeordneten Daten zu Mittelwerten, Verteilungsfunktionen usw. verdichtet.
Wir wollen hier keine Unterscheidung zwischen speziellen Meßmonitoren und allgemeinen Meßmitteln, die vor allem der Abrechnung dienen, machen.

Für ESER-Anlagen stehen für das Betriebssystem OS/ES das SMF /8/ mit spezifischen Auswertungsprogrammen, das GTF /9/ mit dem GTFMON /10/ als Auswertungsprogramm und LLX /11/ zur Verfügung.

## 2.3. Leistungsmonitor EAMON

Die oben angeführten Meßmittel genügen nicht allen Anforderungen, die für bestimmte Leistungsbewertungen erforderlich sind. Deshalb wurde in Zusammenarbeit zwischen der Technischen Universität Dresden und dem Datenverarbeitungszentrum Dresden ein Leistungsmonitor EAMON als Laborprogramm für OS/ES implementiert, erprobt und an einer ESER I-Anlage genutzt.

Das Hauptziel von EAMON bestand darin die E/A-Bedienungszeiten der einzelnen E/A-Geräte: Magnetplattengeräte (MPS7), Magnetbandgeräte (MBS), Paralleldrucker (PD) usw. meßtechnisch unter Umgehung einer genauen Uhr zu erfassen. Für ein Magnetplattengerät setzt sich die E/A-Bedienungszeit aus drei Anteilen zusammen: aus der Positionierzeit $T_p$, aus der Suchzeit $T_s$ und aus der Transferzeit $T_t$ (siehe Abb. 4). Bei den anderen Geräten fallen Positionierzeit und Suchzeit weg.

Die Grundgedanken von EAMON bestehen

- in einer getrennten Bestimmung der Positionierzeit $T_p$, der Suchzeit $T_s$ und der Transferzeit $T_t$
- in einer indirekten Bestimmung der Zeiten $T_p$, $T_s$ und $T_t$ über die Erfassung der Geräteadressen , Zylinderadressen, Datenträgernamen und Blocklängen der physischen Sätze und
- in der Nutzung der Eichkurve für den Zusammenhang zwischen Zylinderabstand k und Positionierzeit $T_p$.

### Erfassung und Aufzeichnung einer Ereignisfolge (Trace) in EAMON

Nach jeder E/A-Unterbrechung erfolgt der Einsprung in das Erfassungs- und Aufzeichnungsprogramm von EAMON. Dieses Programm erfaßt die Ereignisinformation, die dem Ereignis E/A-Unterbrechung zugeordnet ist und zeichnet diese in einem Monitorsatz auf Magnetband auf. Jedem Ereignis ist somit eine Ereignisinformation zugeordnet. Diese besteht aus der Geräteadresse, der Zylinderadresse, der Blocklänge, dem Datenträgernamen und der Uhrzeit. Die Uhrzeit dient nicht zur

Berechnung der E/A-Zeiten, sondern kann für die Zeitsynchronisation mit anderen Meßmitteln (z.B. SMF) genutzt werden. Es wird somit eine Folge von Ereignisinformationen in der zeitlichen Reihenfolge des Auftretens der Ereignisse aufgezeichnet. Diese Folge heißt Ereignisfolge (Trace).

Bestimmung der Eichkurven für die Positionierzeiten

Während die Bestimmung der Suchzeit und der Transferzeit unproblematisch ist, müssen für die Bestimmung der Positionierzeit Eichkurven aufgenommen werden. Hierzu wurden 2 Eichprogramme für ESER I und ESER II und ein Interpolationsprogramm implementiert. Mit diesen Eichprogrammen wurden 25 Magnetplattengeräte:

5 7-MByte-Plattengeräte
16 29-MByte-Plattengeräte
3 100-MByte-Plattengeräte
1 200-MByte-Plattengerät

ausgemessen, wobei insgesamt mehr als 100 Eichkurven für alle möglichen Zylinderabstände tabellarisch aufgenommen wurden. In der Abb. 5 ist für ein Magnetplattengerät vom Typ EC 5061 (29 MByte) die gemessene Abhängigkeit der Positionierzeit $T_p$ vom Zylinderabstand k graphisch aufgetragen. Die Eichkurve des Zusammenhanges

$$T_p = T_p(k) \qquad (5)$$

wird im Auswertragsprogramm des EAMON genutzt.

Statistische Verdichtung in EAMON

Wir beschränken uns hier auf die Verdichtung der Zylinderadressen aus der Ereignisfolge. Hierbei gibt es zwei Aufgaben der Verdichtung

- die Schätzung der Zugriffswahrscheinlichkeiten $p_i$ zu den Zylindern i (i=1,...,N) mit N als Zylinderzahl eines Plattenstapels und
- die Schätzung der Zylinderabstandswahrscheinlichkeiten $P_k$ für die Zylinderabstände $k = |i-j|$, wenn i die augenblickliche Zylinderadresse und j die vorangehende Zylinderadresse in

der Ereignisfolge sind.
Durch die Schätzung der Zugriffswahrscheinlichkeiten $p_i$ zu den Zylindern i können bei Kenntnis der Dateiverteilung über die Zylinder die Zugriffshäufigkeiten zu den Dateien bestimmt werden. Das wird später noch genutzt.
Die Schätzung der Zylinderabstandswahrscheinlichkeiten $P_k$ kann für die Bestimmung der mittleren Positionierzeit $\bar{T}_p$ genutzt werden, wobei gilt

$$\bar{T}_p = \sum_{k=0}^{K} P_k \, T_p(k). \qquad (6)$$

In dieser Formel werden die $P_k$ aus den Zylinderadressen der Ereignisfolge in EAMON und die $T_p(k)$ aus einer abgespeicherten Eichtabelle der entsprechenden Geräteklasse, die vorher über ein Eichprogramm bereitgestellt wurde, gewonnen. In der Abb. 6 ist die aus dieser Messung gewonnene Zylinderabstandswahrscheinlichkeitsverteilung eines Magnetplattenstapels angegeben.

## 3. Nutzung der Leistungsbewertung für die Leistungsverbesserung

Gegenwärtig ist die Leistungsverbesserung im Rechenbetrieb die Hauptrichtung der Anwendung der Leistungsbewertung.
Die wichtigsten Möglichkeiten der Leistungserhöhung auf der Grundlage der Leistungsbewertung bestehen in

- einer geeigneten Wahl der gerätetechnischen Rechnerkonfiguration
- einer geeigneten Planung der Betriebssystemgenerierung
- einer geeigneten Wahl der Start- und Auswahlprioritäten für die Job- und Taskabfertigung und
- einer geeigneten Mischung der Arbeitslast bei der Maschinenbelegung.

### 3.1. Optimale Verteilung der Systemdateien

Nachfolgend wollen wir uns auf ein Beispiel der Leistungserhöhung beschränken, das aus der optimalen Verteilung der Systemdateien über die Plattenstapel und Zylinder der Platten-

stapel besteht. Dieses Beispiel gehört zur geeigneten Planung der Betriebssystemgenerierung.
Die Ziele der optimalen Verteilung der Systemdateien über die Systemplatten bestehen im Lastausgleich zwischen den geteilt genutzten Ressourcen (siehe /5/) und im Erreichen einer möglichst geringen mittleren Positionierzeit der Systemdateizugriffe.
Im konkreten Fall kann diese Aufgabe als Umverteilung der Systemdateien gegenüber einer bereits vorhandenen Verteilung der Systemdateien über die Systemplatten aufgefaßt werden. In diesem Fall bilden die gegenwärtige Verteilung der Systemdateien und die Zugriffshäufigkeiten zu den einzelnen Systemdateien den Ausgangspunkt der Untersuchung. Diese Informationen können durch maschinelle Erfassung der Dateiverteilung aus dem VTOC und durch Verdichtung der E/A-Ereignisaufzeichnung des EAMON in den Zugriffshäufigkeiten $p_i$ zu den Zylindern und damit zu den Dateien gewonnen werden. In Abb. 7 ist die Verteilung der mittleren Zylinderpositionierhäufigkeiten über die Zylinder der beiden Systemplatten DP 9851 und DP 9852 aufgetragen. In dieser Abbildung wurde aus Darstellungsgründen für die einzelnen Dateien eine Gleichverteilung angenommen. Die Messung, aus der diese Darstellung gewonnen wurde, enthält die reale Verteilung über die Zylinder auch innerhalb der Dateien.
Auf der Grundlage dieses Meßergebnisses kann eine optimale Verteilung der Systemdateien vorgenommen werden. Außerdem ist noch das in Abschnitt 2.1 angegebene Beispiel eines Verhaltensmodells für die Abhängigkeit der Rechnerleistung von der Verteilung der Systemdateien über die Plattenstapel und Zylinder der Plattenstapel (siehe Abb. 3) zu nutzen. Auf der Grundlage dieses Modells ergibt sich der aus folgenden zwei Lösungsetappen bestehende Lösungsweg:
- aus der optimalen Verteilung der Systemdateien zwischen den verschiedenen Systemplatten, die zum annähernden Lastausgleich zwischen den Systemplatten führt (siehe /5/, /6/) und
- aus der optimalen Verteilung der Systemdateien jeder System-

platte über die Zylinder, die als Ergebnis einer Positionierzeitverkürzung gegenüber der vorhandenen Systemdateiverteilung hat (siehe /7/).

Nachfolgend werden beide Etappen kurz behandelt und die dabei zu erwartenden Leistungsverbesserungen geschätzt.

## 3.1 Optimale Verteilung der Systemdateien zwischen den Systemplatten

Als Grundlage für die optimale Verteilung der Systemdateien müssen die Zugriffshäufigkeiten $p_i$ zu den Systemdateien gemessen werden. Unter Nutzung dieser Meßgrößen $p_i$ kann dann die Zuordnung der Systemdateien zu den Plattenstapeln erfolgen. Dabei erfolgt die Zuordnung nach der Regel "die Datei mit der größten Zugriffshäufigkeit zuerst" (GZ-Regel). Bei der GZ-Regel werden die Systemdateien ihrer Zugriffshäufigkeit entsprechend in einer Tabelle geordnet, wobei die Datei mit der größten Zugriffshäufigkeit an der Spitze steht (siehe Tabelle 2). Aus dieser Tabelle werden die ersten r Dateien auf die r Systemplatten aufgeteilt. Die Platte mit den wenigsten Zugriffen, erhält dann die nächste Systemdatei zugeteilt. Anschließend erhält wiederum die Platte mit den wenigsten Zugriffen die nächste Datei zugeordnet usw. Dieser Zuteilungsprozeß ist abgeschlossen, wenn alle in der Liste stehenden Systemdateien einem Plattenstapel zugeordnet sind. Falls einer der Plattenstapel während des Zuteilungsprozesses zylindermäßig voll ausgelastet ist, dann werden von diesem Zeitpunkt an nur noch den anderen Systemplatten Dateien zugeordnet. In der Tabelle 3 sind die Ergebnisse der Umverteilung bezüglich der Zugriffshäufigkeiten zu den zwei Systemplattenstapeln DP 9851 und DP 9852 angegeben. Außerdem werden zum Vergleich die gegenwärtigen Zugriffshäufigkeiten zu diesen Systemplatten mit in die Tabelle eingetragen. Es zeigt sich, daß einerseits die gegenwärtigen Zugriffshäufigkeiten mit 36% zu 64% erheblich von einer ausgeglichenen Lastverteilung zwischen den Systemplatten

abweichen und daß andererseits durch die GZ-Regel im vorliegenden realen Fall mit den optimalen Zugriffshäufigkeiten 51% zu 49% eine gute Annäherung an eine ausgeglichene Lastverteilung zwischen den Systemplatten erreicht wird.

## 3.2. Optimale Verteilung der Systemdateien über die Zylinder der Plattenstapel

Hier besteht das Ziel der optimalen Verteilung in der Minimierung der mittleren Positionierzeit $\overline{T}_p$ beider Systemplatten, wobei für eine Systemplatte gilt

$$\overline{T}_p = \sum_{i=0}^{\xi} \sum_{j=0}^{\xi} p_i p_j T_p(|i-j|). \qquad (7)$$

In diesem Ansatz wurde statistische Unabhängigkeit der Zugriffe zu den Systemdateien angenommen. Die optimale Form der Verteilung der Zugriffhäufigkeiten über die Zylinder ähnelt einer Glockenkurve (siehe Abb. 8).
Ein Lösungsweg dieser Aufgabe, der ein quasioptimales Ergebnis liefert, besteht aus folgenden zwei Schritten. Als erstes wird jeder Datei des betrachteten Plattenstapels eine Ordnungszahl k nach der Größe der Zugriffshäufigkeit pro Zylinder $p_i$ (der Datei $D_i$) zugewiesen. Anschließend werden die Dateien von innen nach außen nach einer der Anordnungen

$$N-1,\ldots 5,3,1,2,4,\ldots N$$

oder

$$N,\ldots,4,2,1,3,5,\ldots,N-1$$

über die Zylinder des Plattenstapels verteilt. Hierbei ist N die Zahl der Systemdateien des betrachteten Plattenstapels. Der Bestimmtheit halber wurde N gerade angenommen.
Dieser Lösungsweg wurde auf die beiden Plattenstapel DP 9851 und DP 9852 angewandt, nachdem vorher die optimale Verteilung der Dateien zwischen den beiden Plattenstapeln vorgenommen wurde. Im Ergebnis dieser Neuverteilung der Dateien wurden die in Abb. 9 angegebene Verteilung der mittleren Zylinderpositionierhäufigkeiten über die Zylinder der beiden Plattenstapel erhalten. Die mittleren Positionierzeiten zu jeder der beiden Platten und deren Mittelwerte über beide Platten sind für die

gegenwärtige Systemdateiverteilung und für die durch Optimierung erreichte Systemdateiverteilung in der Tabelle 4 angegeben. Daraus folgt eine Verkürzung der mittleren Positionierzeit beider Platten um 4ms. Bei einer Umrechnung über die Zahl der Systemplattenzugriffe für eine Meßzeit von 12h ergibt das eine Verkürzung der Positionierzeit von insgesamt 47min.

Literaturverzeichnis

/1/ Arato,M. u.a.: Statistische Methoden der Bewertung der Arbeit von Betriebssystemen (russisch) in Selected Papers on Operating Systems Budapest (1978)

/2/ Svobodova,L.: Computer Performance Measurement and Evaluation Methods: Analysis and Applications. Elsevier Computer Science Library New York, Oxford, Amsderdam (1976)

/3/ Ferrari,D.: Computer Systems Performance Evaluation. Prentice-Hall, Inc. . Engleword Cliff, New Jersey (1978)

/4/ Ferrari, D.: Performance of Computer Installations Evaluation and Management. North-Holland Published Company. Amsterdam, New York, Oxford (1978)

/5/ Bergholz, G.: Zur Leistungsanalyse von Rechnersystemen der Stapelverarbeitung, TU Dresden WBZ MKR/IV Hert 43/80

/6/ Bergholz,G.: Leistungsanalyse und Leistungserhöhung von Rechenanlagen. edv aspekte Berlin (1982) Heft 2

/7/ Kogan, Ja., A. u.a.: Erfahrungen der Messung und Anpassung des Betriebssystems OS/ES (russisch) Programmirovanije Moskau (1979) Heft 1

/8/ Erfassen der Ressourcenbenutzung OS/ES (C 5233-0001) Systemunterlagendokumentation. VEB Kombinat Robotron

/9/ Verfolgungshilfen OS/ES. (C 5393-0001-1 E0030) Systemunterlagendokumentation. VEB Kombinat Robotron

/10/ OS-Leistungsmonitor GTFMON, Nutzerhandbuch. VEB Rechenzentrum Karl-Marx-Stadt

/11/ Beschreibung des Programms LLX OS/ES. VEB Kombinat Robotron

Tabelle 1: Prozentuale Auslastung verschiedener festzugeordneter Ressourcen

| Geräteklasse | HS | MP6 | MB6 | PD |
|---|---|---|---|---|
| Konfiguration | 512 kByte | 12 Geräte | 16 Geräte | 3 Geräte |
| Auslastung in % | 47,8 | 35,8 | 28,3 | 22,4 |

Tabelle 2: Zugriffshäufigkeiten zu den wichtigsten Systemdateien

| Datei | absolute Zugriffszahl | absolute Zugriffszahl Zylinderzahl |
|---|---|---|
| SYS1.SYSJOBQE | 246 322 | 4 105 |
| SYS1.SVCLIB | 209 278 | 8 049 |
| SYS1.LINKLIB | 152 166 | 7 608 |
| SYS1.COMPILE | 35 467 | 548 |
| SYS1.CTLG/VTOC | 27 895 | 27 895 |
| SYS1.MRLINK | 12 246 | 1 113 |
| SYS1.MANX | 6 765 | 1 691 |
| SYS1.SORTLIB | 3 812 | 953 |
| SYS1.PL1LIB | 3 719 | 619 |
| SYS1.SYSVLOGX | 2 045 | 227 |
| SYS1.COBLIB | 1 950 | 1 950 |
| SYS1.PL1OPT | 858 | 57 |
| SYS1.LOGREL | 252 | 126 |
| SYS1.PROCLIB | 101 | 51 |
| SYS1.MRPROC | 76 | 25 |

Tabelle 3: Vergleich der Zugriffshäufigkeiten der Umverteilung mit den gegenwärtigen Zugriffshäufigkeiten

| Platten-stapel | Zugriffshäufigkeiten | | | |
|---|---|---|---|---|
| | gegenwärtig | | optimal | |
| | absolut | relativ | absolut | relativ |
| DP9851 | 252 446 | 0,3576 | 362 630 | 0,5136 |
| DP9852 | 453 549 | 0,6424 | 343 365 | 0,4864 |
| Σ | 705 995 | 1,5000 | 705 995 | 1,0000 |

Tabelle 4: Ergebnisse der Umverteilung

| Plattenstapel | mittlere Positionierzeit $\bar{T}_p$ [ms] | |
|---|---|---|
| | gegenwärtig | optimal |
| DP9851 | 25,25 | 29,11 |
| Dp9852 | 42,72 | 36,02 |
| ø | 36,48 | 32,45 |

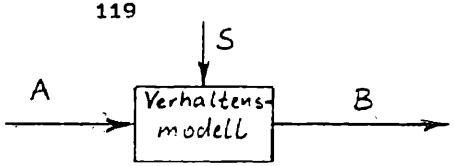



Abb. 1: Blockschema für das Verhaltensmodell eines Rechner-systems

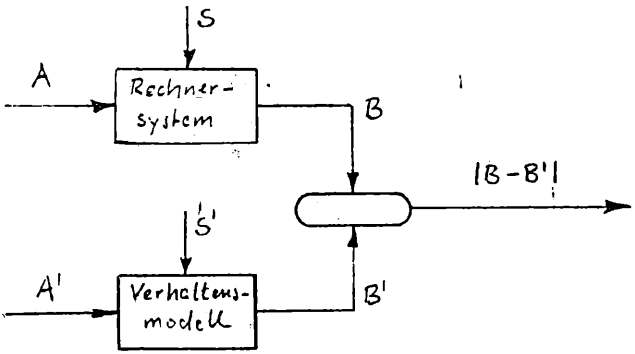



Abb. 2 Zur Untersuchung der Adäquatheit eines Verhaltensmodells

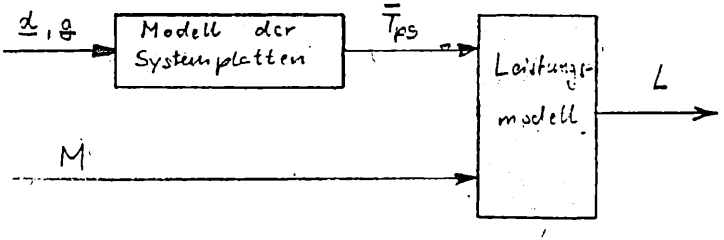



Abb 3: Abhängigkeit der Rechnerleistung von der Verteilung der Systemdaten auf die Plattenstapel und Zylinder

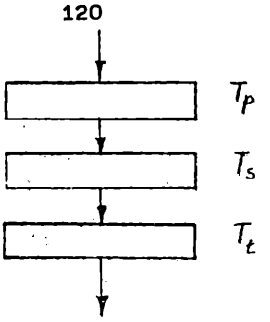



Abb 4 Bestandteile der E/A-Bedienungszeit

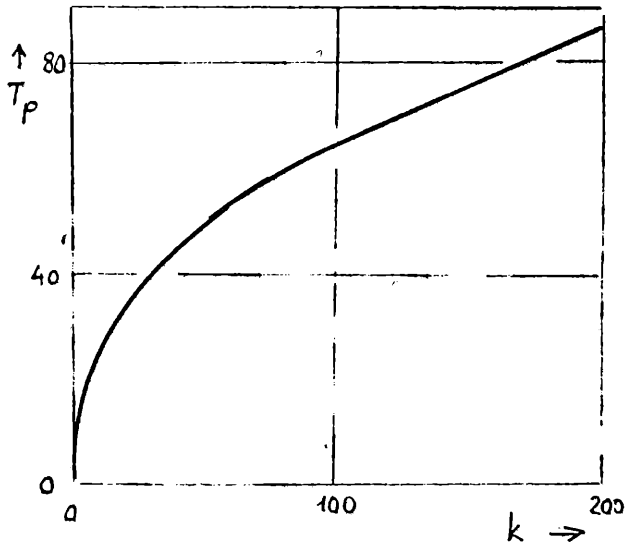



Abb. 5: Abhängigkeit der Positionierzeit vom Zylinderabstand (Eichkurve)

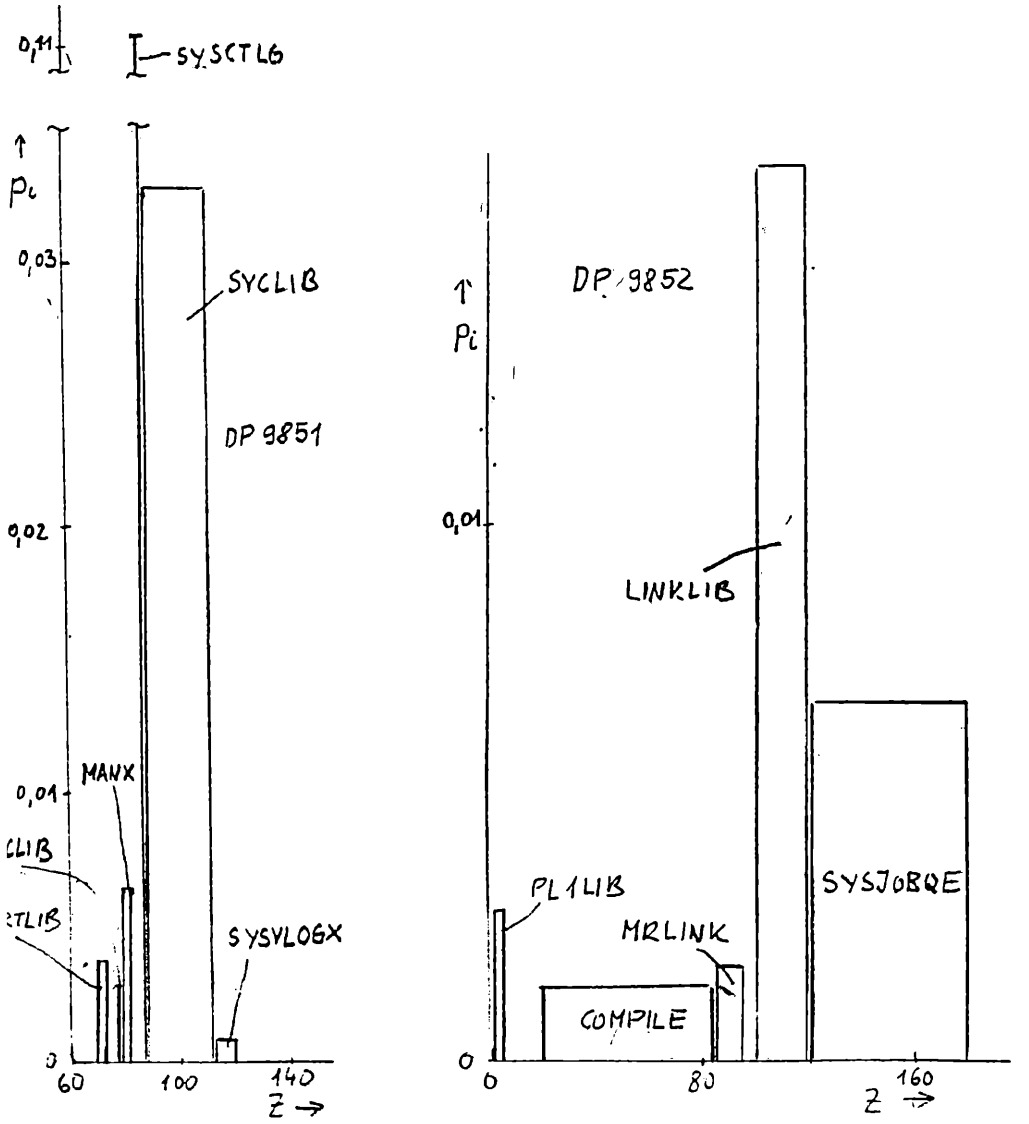



Abb. 7 Verteilung [illegible] pro [illegible]
Datei

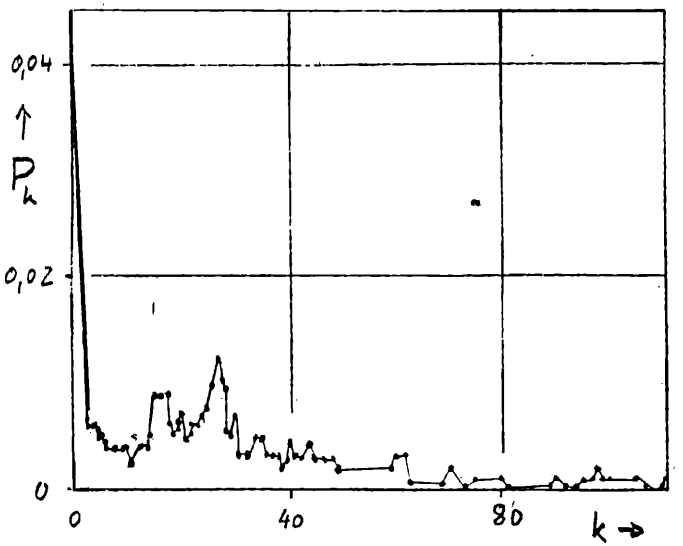



Abb. 6 Verteilungsfunktion der Zylinderabstände einer Systemplatte

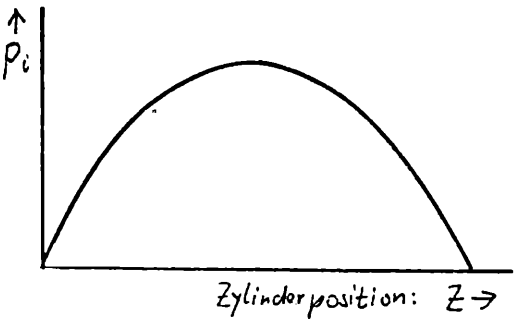



Abb. 8 Optimale Form der Zugriffsverteilung

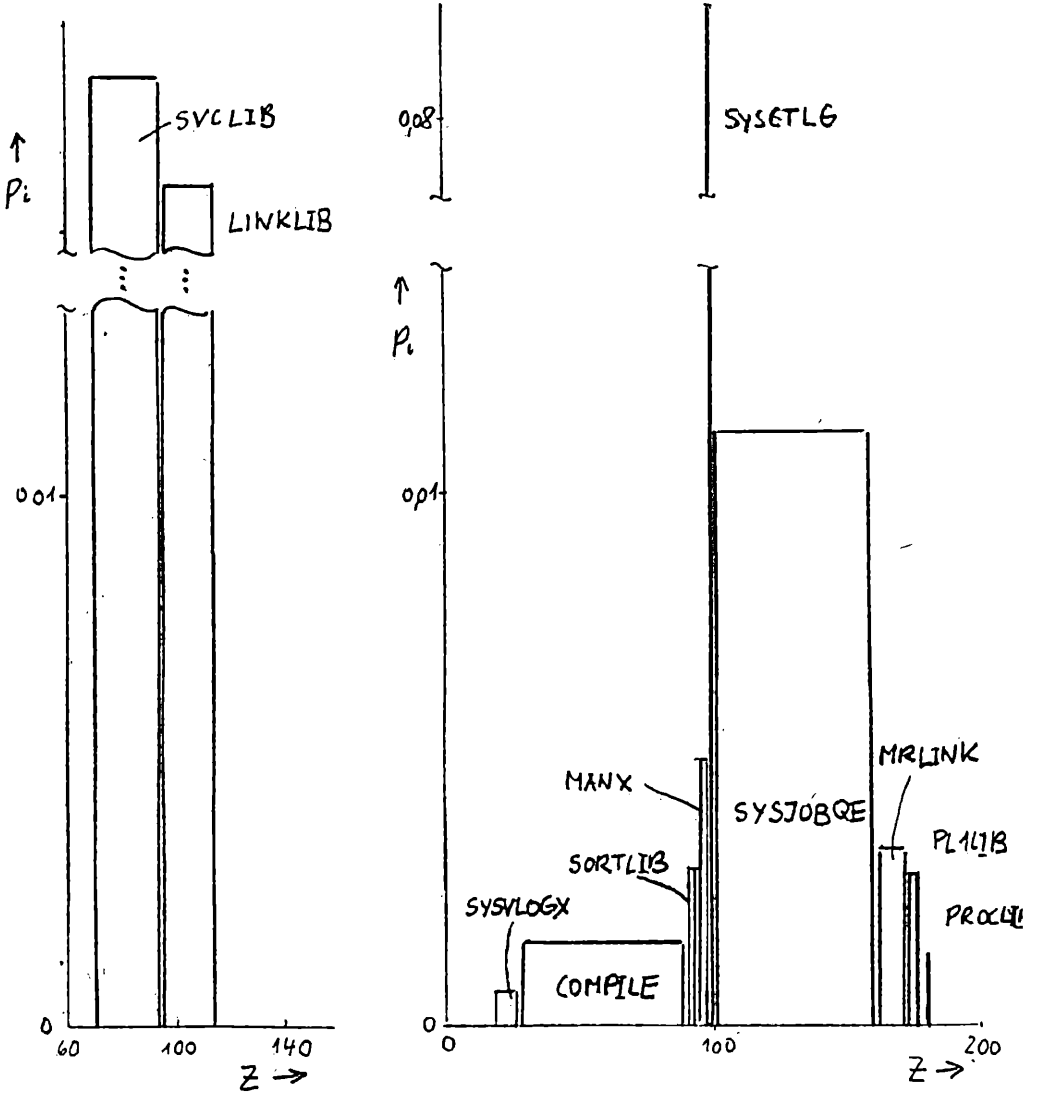



Abb. 9 Verteilung der mittleren Zugriffspositionshäufigkeiten in den Dateien nach der Umverteilung

Bazewicz Mieczysław, Mika Teodor

# User conditions to software functions in distributed processing system /DPS/

## 1. Introduction

Research activities in universities, based upon more and more precise methods and calling therefore for more and more complex observations, are often carried on with the use of computer systems supporting not only calculations of scientific workers, but also data gathering, segregating and analyzing. Computer applications in the research process cause tasks to be solved by the computer system designers, concerning the selection of hardware and logical architecture being appropriater to research work. The hitherto unsolved problem is to develop methods of adapting computer system architecture to the needs of research works carried on in universities.

In our paper a method of software and hardware adapting to user needs will be presented of the computer user point of view, which differs of the point of view of software and hardware producer or owner. The approaches to computer system adapting performed up to now have taked into consideration mainly the computer throughput and the maximal load during user job execution. Our approach differs of them in that the main criterion of configuration adapting is the best service of user, which requirements and needs result from specific functions in research activities proper to his concern. The other criteria fulfill only auxiliary rules such as boundaries, which should be exceed by designing new configurations based upon new technologies.

Into our mainly consideration is the development arising in technology and system organization and its influence into the possibility to fulfill users requirements. From this result some

changes in methods of user system designing which are the second theme of our concern.

In uniprocessor systems, architecture adapting was performed only within a limited scope, considering the type of hardware and the necessity to use complex program mechanisms protecting against mutual interference of different calculation processes executed in parallel. Uniprocessor systems imposed, in turn, constraints concerning users' requirements, forcing the latter into the possibilities offered by uniprocessor. The problems of hardware and software choice, taking into considerations users' requirements without imposed constraints, can be more efficiently solved by designing, for those requirements, more sophisticated systems composed of many processors, such as local computer networks.

So, the adaptation of computer system architecture to users' requirements is a complex problem, as it depends upon many different factors: upon the characteristics of the user and his job, the requirements reffering to the method of job processing, and it depends also upon the elements, which the computer system can be built from and the use of which permits to fulfill the users' requirements.

The modern technologies provide the design of multiprocessor systems, remote networks and some medial solutions such as local area network /LAN/.

The purpose of multiprocessor systems is the increasing of computer system performance for such applications as the computations for weather - forecast. The purpose of remote network is the increasing of access to large computer systems for users situated in wide space. The local area network has an elastic structure, which is suitable to satisfy user needs.

How great importance have the new technologies proves the last IFIP conference in Florence /Italy/, where new technologies have been presented for local area network building. Among other, the

Advanced Micro Devices, Inc. has announced the introducing to the market some elements based on VLSI, complexly solving the building problems of local area networks. In that solution the local area network was divided into two subnetworks: the frontend network and the backend one.

The frontend network serves to bind terminals to CPU's, and the backend network links the CPUs with high speed auxiliary memories.

Different bit rates are required for both networks. The frontend node based on AMDelements is similar to Ethernet node and support bit rates from 50 kbs to 200 kbs. The backend node support bit rates from 10mgbs to 1 gbs. This fulfill the ANSI recommendation, which defines the necessary bit rate at a level of 50 mgbs, but this bit rate may not be justifiable in all networks.

The high speed node for the backend network is a programable device, which requires about 70 chips, and process a protocol for data exchange between large CPUs and high speed disks.

In uniprocessor systems adapting the architecture to users' requirements is restricted by the possibility of hardware selection. A certain freedom of choice was obtained only what concerns the software, which, also, was restricted by the hardware characteristics. In distributed systems more degrees of freedom exist when solving that problem. The choice concerns, indeed, depending on user's requirements:

- the number and kind of system processors,
- the dedication of processors to determined user's jobs,
- the user's job division into processes performed in different parts of a distributed computer system.

However, the choice of hardware and software elements in the case of a higher number of degrees of freedom sets more complex problems for the designer. The choice of an effective solution, from among a wide specter of them, requires to state precisely the criteria of system architecture adapting to users' requirements,

the most obvious one being the criterion of power /performance/ balancing according to the requirements.

An approach to such a method of solving the problem of adapting architecture to users' requirements, based upon a univocal representation of the processes of a real object in a computer system architecture is the subject of this paper. The point of the approach consists in trying to obtain, in designing, a possibly adequate representation of the environment by the system of servicing those requirements. That approach will be discussed upon an example of experiment planning and controlling in a university laboratory.

## 2. A method of system architecture adapting to users' requirements

A number of trials of adapting distributed system architectures, namely the ICN /Interuniversity Computer Network/ architecture, to users' requirements in universities for research work supporting, was presented in [1]. The problem of developing an efficient method of distributed system designing is considered to be a problem of formulating relations between two vectors containing variables determining the information processes from the user's point of view, and variables describing the inner structure of the system assuring the required user services.

The users' requirements and the external environment constraints are described by the vector $X = (X_1, X_2, \ldots, X_n)$ of primary variables, this vector determining the user properties or services performed by the computer system. The $X_i$ variables concern: features and constraints of users, characteristics of the job to be processed, precision, reliability and secrecy, and other constraints imposed by the environment, such as hardware and software availability, financial means, etc.

The parameters of the system for users' requirement servicing are described by the vector $Y = (Y_1, Y_2, \ldots, Y_m)$ of secondary variables, that vector determining the inner structure of the computer system. The $Y_j$ variables concern logic structure of the system,

system nodes, routing, organization into subsystems and hardware and software elements. Those variables are grouped depending upon their influence on the system /full or partial influence on a partial influence on a particular element/, and upon subsystem activeness /data processing and transmission/.

Each of the primary variables depends upon a number or all secondary variables, i.e. $X_i = x_i(\tilde{Y})$, where $\tilde{Y}$ is the vector of some or of all secondary variables. In order to evaluate the solutions obtained in the design synthesis process, the variables of a performance or evaluation vector $P = (P_1, P_2, \ldots, P_s)$, are introduced, where the $P_i$ variables, for $i = 1, \ldots, s$ are e.g. waiting time, error rate etc. The evaluation basis is the user job handling process, and the evaluation criterion is efficiency, job handling mean time. That evaluation permitted to prove the usefulness of the method for selecting the organization and number of nodes in the communications subnet.

The method has, however, another significance and importance in the case of Local Computer Networks /LCN/, where the nodes have a different destination and where there is no communications subnet.

## 3. Laboratory environment of Local Computer Network

The problem of computer system architecture adapting to user requirements will be presented, considering 15 existing important laboratories at the Technical University of Wrocław. Some of those laboratories are equipped with minicomputer systems connected trough module systems of measuring and control automation units of the CAMAC type to the experimental object. In such a laboratory automatic collection of measuring data of an experiment carried out can be performed, as well as a relatively simple processing of those data in order to determine the character of the experiment and to verify the hipotheses formulated by the experiment goal. In practice, the laboratory experiment includes not only experiment control, measuring and analysis of measurement data,

but also the process of experiment planning, taking into consideration the results and conclusions from the previous experiments carried out. Thus, a computer system supporting experiment planning and control should permit to store data and results from previous experiments in the form of an experimental data base, It is not possible on economic grounds, to equip every laboratory with such computer systems, so the solution of the problem of architecture design is based upon computer network.

Let us nearly examine the processes of experiment planning and running in a university laboratory, considering the case of a mechanical laboratory [2]. The following typical processes, for experiments in different domains of research, can be isolated from among all experiment planning and running processes in a university research laboratory:

P1. Setting the measuring equipment, consisting in determining the measuring range and calibrating measuring instruments. That process is generally hand performed, although it can be computer supported when setting up the initial values /e.g. zero setting/ of instruments by means of CAMAC channels.

P2. Setting up external conditions forcing the required behaviour of the experiment object. Those conditions are for example an appropriate value opening or closing in chemical experiments, or applying an exciting force in strength experiments.

P3. Measurement data input through measuring systems /indicators, signal converters etc..../ in a determined time interval for dynamic experiments, or on the basis of N-tuple measurements in static experiments.

P4. Preparatory processing of the measurement data by assigning suitable physical values to electrical values obtained by measuring , and calculating means values of a series of measurements, and the standard deviations or variances.

P5. Pre-evaluating the obtained measurement data, i.e. evaluating the likelihood, e.g. by comparing the standard deviations with the maximum set value. That evaluation permits to eliminate the

so-called "big" errors consisting in data input from a measuring point out of calibration or a broken down one.

P6. Measurement data processing according to a model formulated by the experimenter, and obtaining in this way the basic relations existing in the experiment, e.g. in the form of coefficients of determined linear function, exponential function, etc.

P7. Comparing the obtained results with data concerning experiments previously carried out and stored in the data base. Decision making concerning model change or experiment repetition under other external conditions.

P8. Dialogue carrying on with the user, who forces the beginning of the particular experiment phases, who is informed on partial results, and who makes decisions concerning the directions of experiment running or the method of measurement result analysis.

The user signals the beginning of processes P2 and P3, and he is informed by processes P5 and P8 about evaluation results on the basis of which he signals again the beginning of measurements. The user decides also about the selection of the model or relations used in process P6, and about running the whole experiment, by changing the external conditions or the disposition of measuring points.

Process P2 should be performed, especially for dynamic experiments requiring to control the external conditions in a closed loop, by a very fast control system, thus by a control system or simple processor which permits to produce and to emit control signals within a short time.

Process P3 should be performed, for a high number of measuring points, in a processor with wide possibilities of multiplying ways of data collection.

Process P4 should be performed in a processor permitting to execute fast, simple calculations on a wide quantity of data.

Process P5 can be performed in a simple processor, i.e. a processor permitting to execute simple calculations on a somewhat less number of data, /than in process P4.

In order to perform process P6, a processor with higher computing power is required for executing somewhat more complex calculations, but on aggregated data, thus on a smaller number of data.

For performing process P7, mass stores are necessary to store in a suitable data base an important as possible number of experimental data, serving as standard data for further experiments.

Process P8 requires a processor with terminal devices for carrying on dialogue.

For each of the discussed processes a processor with different characteristics is required, and it can be pre-assumed that a special processor will be assigned to each process, each processor being the best adapted on to its own process. As there is a need for transmitting information from one process to another one, the mentioned special processors should be interconnected, and as they should be processors with different possibilities, those connections should form a computer network. The small separation of the particular processors makes it possible to design a local computer network.

However, the need for obtaining the automation of the whole laboratory complex, with economically grounded restrictions, causes the distributed system should be designed in such a way that more developed processors should not be multiplied in all laboratories /such elements developing the system being mass stores for process P7 and a set of terminal devices for user-system dialogue in process P8/. Thus, it is intended to connect all the simpler processors performing processes P1 - P6 into a local computer network, and to utilize the ICN computers as more developed systems.

## 4. Laboratory ICN configuration

On the basis of the above - mentioned process analysis, without considering the restrictions imposed by interprocessor communication, the conclusion arise, that a local computer network

destined for laboratory experiment planning and running should be of the star architecture where, in the center the system D is located for the dialogue with the user, system D being connected with system E, in which information is stored in the base of experimental data. At the end of the rays of the star the processors $C_i$ with systems of CAMAC type measuring and control automation are located, with circuits multiplying the measuring ways. Each laboratory should be equipped with such systems. Systems D and E are connected to the Interuniversity Computer Network, considering they have to be big systems permitting to perform, beside processes for experiment planning and running, many other tasks concerning scientific research in the university. Moreover, the separation of both these systems can be important, as the processes being performed are relatively slow, but big size auxiliary store are needed.

Between system D, where process P8 is performed, and systems $C_i$ performing the fundamental control and measuring processes P2 and P3 processors $A_i,\ldots,B_i$, performing processes P4 to P6 are located.

If a separate processor is assigned to each of the processes P4 to P6, such processor being the best adapted one to the particular process, a full suitability of the architecture of many processes and many processors is obtained, that suitability being called homomorphism by Jensen [3]. However, the requirements of the neighbor processors, although different, are somewhat similar, e.g. for performing processes P4 and P5 the processors should fast execute relatively simple calculations, thus both these processes can be installed in a single processor, the more so that the processes are consecutively performed. Two other processors $A_i$ and $B_i$ have been decided to be applied between system D and systems $C_i$, as a result of analysis of the whole of experiment control processes considering the 3 following criteria:

- adapting the processor performance,
- assuring an appropriate response time of different processors,
- minimizing the information volume being transmitted from a process to the next one.

5. Some selection criteria

Requirements concerning computing power have been discussed in point 3. It can be noted that the processing performance for that application was growing for the consecutive processes. Thus a suggestion arose to consider a star architecture, where, in the center, the system with the greatest performance would be located, and in the star rays - the most simple processors.

The response times for a complex application such as experiment control should be different for different stages of that application. Here a time can be considered which goes by from receiving a signal indicating that a physical condition has the value $X - \Delta X$ to the instant of emitting a control signal serving to correct the value $\Delta X$ to level X. So, it is the response time of the control system performing process P2. Another response time is the time which goes by from the beginning of measuring in process P3 to receiving a pre-evaluation of measurement results, obtained from process P5. A reduction of that response time can be achieved by integrating the performing of processes P3, P4, and P5 in a single, bigger system. That fact is one of the arguments against the introduction of a full homomorphism into a distributed system.

A third response time, nearer to the response times in standard time-sharing systems, is the dialogue response time /process P8/ measured e.g. from the instant of model describing to the instant of obtaining the basic relations in process P6.

All the above responses are required after a lapsee of different times. The first time, for fast physical and chemical processes, should not exceed a few microseconds. The second time depends upon the velocity of the process itself in the case of a dynamic experiment. However, as in that time the decision is made on experiment repeating, the time should be fractions of a second. The third response time can be of an order of even a couple of seconds, but that does not interfere with the possibility of obtaining that time from a system connected by a terminal to a remote network.

Between processes P1 - P8 different information is exchanged. From process P3 to process P4 a matrix N x N is transmitted, where N indicates the number of repeated measuring or the number of time interval division of the experiment into instants in which the measurements are executed, and M indicates the number of different measuring points. From processes P4 to processes P5 two or four vectors with length M are transmitted, these vectors including means, variances or standard deviations, and, sometimes also values of maximum and minimum quantities. The same vectors are transmitted to processes P6 which performs the processing of the data vectors obtained from P3, but on the ground of information /model form/ from process P7 storing experimental data files. The data of the running experiment are transmitted to those experimental data, between processes P6 and P7.

## 6. Conclusions

As a result of such analysis, we can demonstrate that some similar processes may be performed in a single system. We obtain, in the same way, a less information volume flow between different processors, and not increased system response time. The architecture adapting of a distributed system /LCN/ to real system requirements, as in the example of the above discussed experiment control system, cannot be achieved only by a logical decomposition of the whole process into simpler, sufficiently different subprocesses from the point of view of their properties, and next by the structural correspondence according to the homomorphism principle of process to processor structures, although that allows to obtain a short response time [4]. Taking into consideration the system performance, system response time and volume of exchanged information, by architecture selecting to user requirements, is indispensable for design with redundance avoiding and definition of some specific rules of information exchange and the kind of synchronization between the LCN nodes.

The method presented in point 2, may be therefore applied not only onto the communication field [1], but also in processing aspects onto LCN, and the vectors: X - of user requirements, Y - of system inner structure, and P - of selection criterion, allow to determine the dependence X=x(Y) in processing element space.

References:

[1] Bazewicz M., Pužman J., The Role of User Job Handling Process in the Computer Network Design Methods. Working Papers of COMNET'81. May 81. Budapest.

[2] Kasprzak W., Lysik B., Computer System in a Research laboratory /in Polish/, Technical University of Wrocław, 1981.

[3] Jensen E.D., The Honeywell Experimental Distributed Processors - An Overview. Computer. Jan. 1978.

[4] Peberdy N.J., Distributed Computer Systems - A Review, Questiones Informaticae, V. 1, No 1, 1979.

Verfassers:

Doz.Dr.Ing. Mieczysław Bazewicz
Dipl.math. Teodor Mika
Technical University of Wrocław, Poland
Institute of Control and Systems Engineering
50-370 Wrocław
Wybrzeże Wyspiańskiego 27

Heymer, Volker

# Tendenzen der Datenkommunikation

## 1. Einleitung

Die elektronische Informationsverarbeitung hat sich in allen Bereichen der Volkswirtschaft einen festen Platz erobert. Sie ist ein integraler Bestandteil in Leitung und Planung, Entwicklung und Produktion geworden.
Durch die breite Einfuehrung von Mikroelektronik und Robotertechnik erweitert sich ihr Anwendungsgebiet betraechtlich.
Der Frage des Zugriffes zu EDV-Systemen und ihrer Standortbestimmung kommt unter dem Gesichtspunkt einer effektiven Nutzung der Rechenkapazitaet eine grosse Bedeutung zu. Die gegenwaertige Situation ist dadurch charakterisiert, dass der Nutzer seine Aufgaben zum Rechner bringen muss (Abb. 1a) bzw. ueber Terminals fest mit einem bestimmten EDV-System verbunden ist (Abb. 1b).

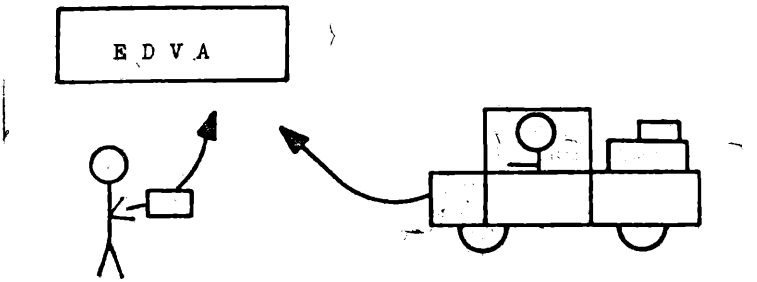



Abb. 1a: Klassischer Zugriff zu EDV-Systemen

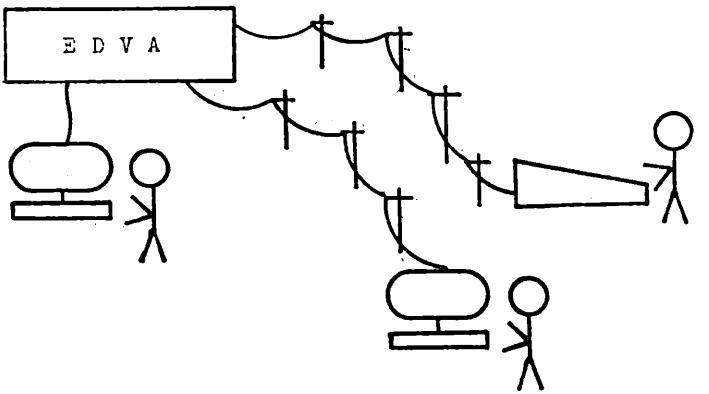



Abb. 1b: Zugriff zu EDV-Systemen ueber lokale und entfernte Terminals

Obwohl der zweite Fall fuer den Nutzer eine deutliche Verbesserung seiner Zugriffsbedingungen schafft, bleiben zwei Probleme offen:

- der Zugriff des Nutzers zu anderen Rechenanlagen, die aufgrund von Spezialressourcen einschliesslich Programmbibliotheken fuer bestimmte Aufgaben besser geeignet sind als die eigene Rechenanlage und
- die geringe Ausnutzung der verwendeten Verbindungsleitungen und damit die oekonomisch bedingte Unmoeglichkeit der Bereitstellung dieser Art des Zugriffs zu EDV-Systemen fuer jede Anforderung.

Die Loesung fuer diese Probleme liegt in Analogie zur vergleichbaren Situation bei der Energieverteilung oder im Fernsprechnetz in der Schaffung von Verbundsystemen, Daten- und Rechnernetzen (Abb. 2), an denen gegenwaertig international in grosser Breite gearbeitet wird.

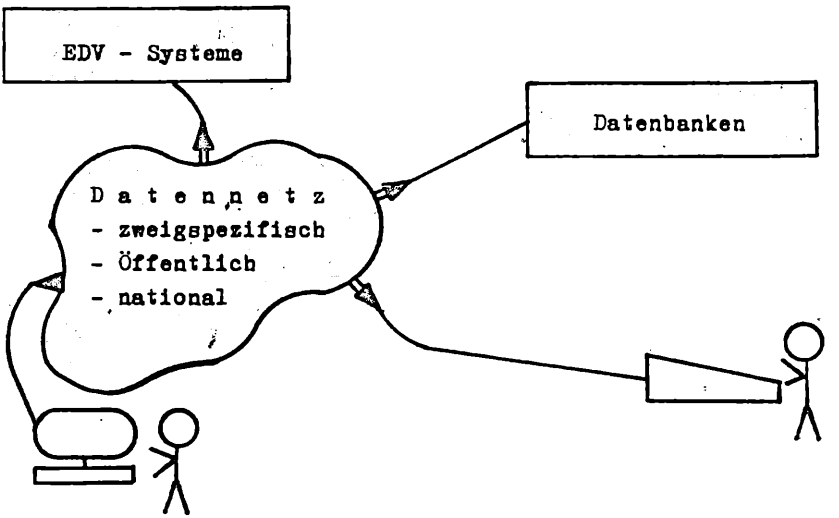



Abb. 2: Verbund von EDV-Systemen und Terminals ueber ein Datennetz mit standardisierten Schnittstellen

## 2. Internationale Situation auf dem Gebiet der Datenkommunikation

Die Entwicklung der elektronischen Informationsverarbeitung wird gegenwaertig durch zwei Faktoren besonders beeinflusst.

Der erste Faktor ist die Mikroelektronik. Durch sie wird eine bedeutende Kostenreduzierung auf dem Hardware-Sektor erreicht. Damit werden die oekonomischen Voraussetzungen geschaffen, um die Informationsverarbeitung in neuen Bereichen anzuwenden.

Der zweite Faktor ist die Verbindung von EDV-Systemen und Telekommunikationssystemen. Dadurch wurde einerseits der entfernte Zugriff zu Rechenanlagen und deren Verkopplung moeglich und andererseits wurden durch den Einsatz von Mitteln der Rechentechnik in der Nachrichtentechnik neue

Uebertragungsverfahren entwickelt und eine Leistungssteigerung erreicht, die die Voraussetzung fuer die Verkopplung von Rechnern und Terminals zu Rechnernetzen ist.

Moderne Rechnersysteme erbringen eine hohe Verarbeitungsleistung und verfuegen ueber Massenspeicher, auf deren Basis umfassende Informationssysteme aufgebaut werden koennen. Datennetze sind die dazu erforderlichen Transportsysteme, um diese Leistungen zum Nutzer zu bringen bzw. die Daten zu sammeln und zu verteilen.
Durch die Mikroelektronik werden billige Rechnerleistung bzw. Steuerkapazitaeten an den Einzelarbeitsplatz des Wissenschaftlers, Ingenieurs, oder Produktionsarbeiters gebracht. Deren Arbeit ist im wachsenden Masse durch Kooperation oder Verflechtung mit der Arbeit anderer gekennzeichnet. Die dazu erforderliche Verbindung der einzelnen Rechner- bzw. Steuersysteme wird mit Mitteln der Rechnerkommunikation realisiert.
Eine quantitative Erfassung dieser Situation fuer Westeuropa wurde mit der Eurodata'79 - Studie /EUR79/

| Anwendung | % der Zugriffspunkte |
|---|---|
| Software-Entwicklung | 11 |
| wiss.techn. Berechnungen | 17 |
| Zugriff zu Datenbanken | 4,1 |
| Mensch-Mensch-Kommunikation | 18 |
| Allg. Leitung/Verwaltung | 39 |
| Bankwesen | 30 |
| Buchungssysteme | 3,1 |
| Börse | 17 |
| Zugriff zu Patientendaten | 4,4 |
| andere | 4,4 |

Abb. 3: Hauptanwendungen der Datenkommunikation in Westeuropa - Eurodata'79 - Studie

versucht, in der 141000 Zugriffspunkte zu Informationsverarbeitungssystemen ueber Datenkommunikation in 17 Laendern bewertet wurden (Abb. 3).
Waehrend zu Beginn der Rechnernetzentwicklung die Anwendungen Software-Entwicklung, wissenschaftlich-technischen Berechnungen und Buchungssysteme dominierten, ist aus der Abbildung eine noch jetzt anhaltende Tendenz zur raschen Zunahme der Anwendungsfaelle auf den Gebieten der Leitung/Verwaltung, der Geldwirtschaft und der Mensch-Mensch-Kommunikation zu entnehmen.
Die spuerbaren oekonomischen Vorteile, die sich aus dem Einsatz der Datenkommunikation ergeben haben, fuehrten dazu, dass nach Rechnernetzen, die vorwiegend im Bereich der Forschung eingesetzt wurden und selbst Gegenstand der Rechnernetzforschung waren, wie das amerikanische ARPANET /MCWA77/ und das franzoesische CYCLADES /POU73/, bald erste zweigspezifische und dann auch oeffentliche Datennetze und Rechnernetze auf kommerzieller Basis Rechnerkommunikationsdienste anboten.
Auf dem Gebiet der oeffentlichen Datennetze ist dabei besonders das Paketvermittlungsnetz PSS (Packet Switch Service) /MED80/ der Britischen Post hervorzuheben (Abb.4). Die Britische Post spielte und spielt eine Pionierrolle unter den Nachrichtenverwaltungen auf dem Gebiet der Datenkommunikation und deren Anwendung fuer neue Dienste. So ist das System PRESTEL /CHI81/ der Prototyp fuer den neuen Telekommunikationsdienst VIDEOTEX, der von der CCITT empfohlen wird. Er stellt durch Kombination von Fernsehgeraet und Telefonanschluss unter Ausnutzung der Rechnernetztechnologie Dienste bereit, die neben dem Einsatz in der Industrie, ein breites Spektrum der Anwendung von Datenbanken und Rechnersystemen fuer solche Bereiche der Volkswirtschaft, wie Dienstleistungen, Handel und Handwerk sowie fuer Haushalt und Freizeitgestaltung ermoeglichen /HERA82/.

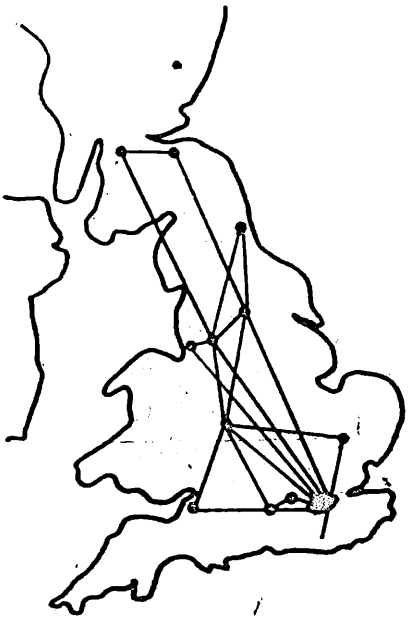


Abb.4: Das Paketvermittlungsnetz PSS der Britischen Post in der Ausbaustufe 1981

In den sozialistischen Laendern lag der Schwerpunkt der Arbeiten zu Rechnernetzen im Bereich der Akademien der Wissenschaften. In den Rechnernetzen DELTA (s. Pkt. 4), CEKOП /MJA78/ und dem Experimentellen Rechnernetz der AdW der Lettischen SSR /YAK78, KIK82/ (Abb. 5) ist die entfernte Auftragsbearbeitung die Hauptanwendung. Die Arbeiten wurden von den Rechnerherstellern aufgegriffen. Ein erstes kommerzielles System liegt mit dem Rechnernetz VNS /RAJ81/ vor. An ein Paketvermittlungssubnetz sind hier Terminals und Rechnersysteme auf denen sich Datenbanken befinden, angeschlossen. Dabei kann von einem beliebigen Terminal zu einer beliebigen Datenbank zugegriffen werden und die Terminals koennen untereinander kommunizieren (Abb. 6).

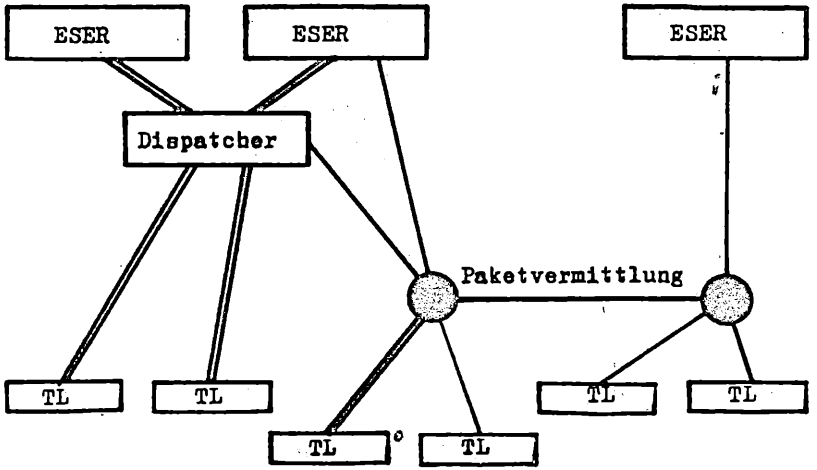



Abb. 5: Experimentelles Rechnernetz der AdW der Lettischen SSR

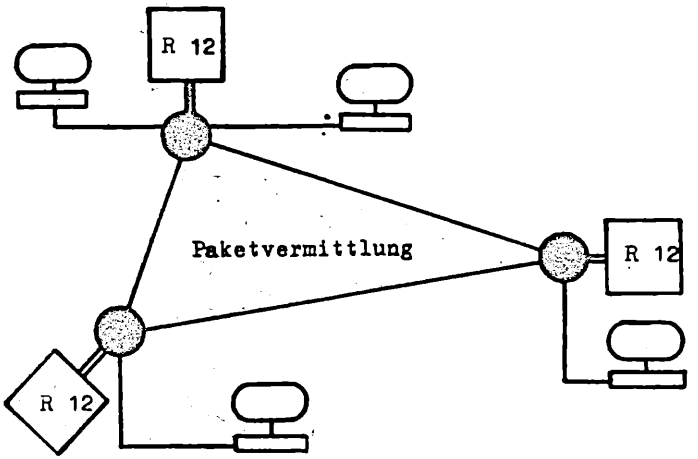



Abb. 6: Videoton Network System VNS

Die wachsende Anwendung der Rechnernetztechnologie hat dazu gefuehrt, dass Datennetze ueber die Laendergrenzen hinaus erweitert wurden, wie die amerikanischen Netze TELENET und TYMNET, und in starkem Masse an internationalen Standards gearbeitet wird.
Die Standardisierungsarbeiten konzentrieren sich
- im Subkomittee 16 des Technischen Komittees 97 der Internationalen Organisation fuer Standardisierung (ISO/TC97/SC16)
- in der Studiengruppe VII des Internationalen Konsultativkomittees fuer Telephonie und Telegraphie (CCITT/Study Group VII)
- in der Gemeinsamen Spezialistensektion Rechnernetze des ESER und SKR (GSS RN) und
- im Technischen Komittee 6 der Internationalen Foerderation fuer Informationsverarbeitung (IFIP/TC6).

## 3. Rechnernetzarchitektur - Verkopplung offener Systeme

Allgemeine Prinzipien zum Verbund von Rechnersystemen zu Rechnernetzen sind von der ISO im Dokument "Data Processing - Open Systems Interconnection - Basic Reference Model" /ISO7498/ (OSI-Modell) fixiert.
Dieser Internationale Standard schafft den Rahmen fuer die Koordination von existierenden und zukuenftigen Standards zur Verkopplung von Systemen. Unter einem System werden dabei eine oder mehrere EDVA, die entsprechenden Programme, peripheren Geraete, Terminals, Bediener, Prozesse, Informationsuebertragungseinrichtungen usw. verstanden, die ein autonomes Ganzes bilden, das Informationen verarbeiten und/oder uebertragen kann. Ein solches System wird als "offen" bezeichnet, wenn es die entsprechenden Standards zur Kommunikation mit anderen Systemen anwendet.
Als Uebertragungsmedien werden Telekommunikationsmedien vorausgesetzt.
Das OSI-Modell beinhaltet nur Probleme der Kooperation zwischen den Systemen und nicht die interne Struktur und Funktionsweise der einzelnen Systeme.

Aufbauend auf Erkenntnissen zur Struktur von Rechnersystemen und den Erfahrungen aus der Entwicklung experimenteller Rechnernetze wurde die in Abb. 7 dargestellte Schichtenarchitektur ausgearbeitet.

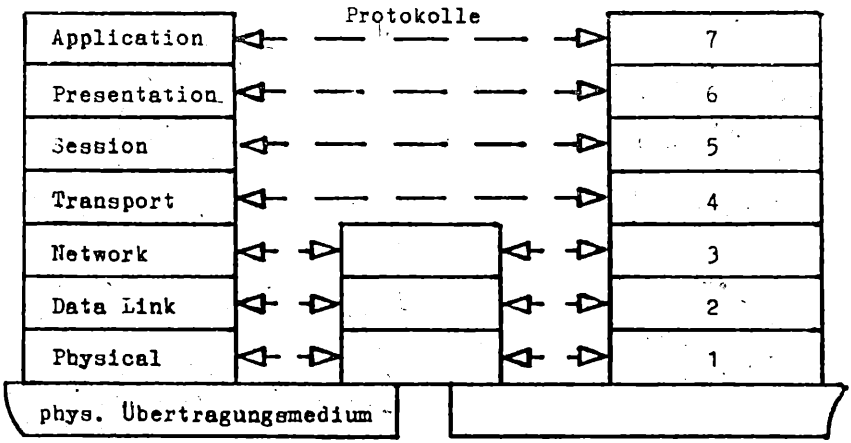



Abb. 7: OSI-Schichtenmodell zur Verkopplung offener Systeme

Sie stellt ein hierarchisches System von Schichten dar, in dem die (n)-Schicht der (n+1)-Schicht einen (n)-Dienst bereitstellt und dazu den (n-1)-Dienst der (n-1)-Schicht in Anspruch nimmt.

Die (n-1)-Schicht besteht aus Subsystemen des gleichen Niveaus in den verkoppelten Systemen.
Der (n)-Dienst wird von (n)-entities der (n)-Schicht bereitgestellt. Sie arbeiten dazu nach dem (n)-Protokoll zusammen. Ein Protokoll ist ein Satz von Regeln und Formaten (Semantik und Syntax), der die Kommunikation zwischen den (n)-entities bestimmt.

Das Zusammenwirken der entities ist in Abb. 8 dargestellt.

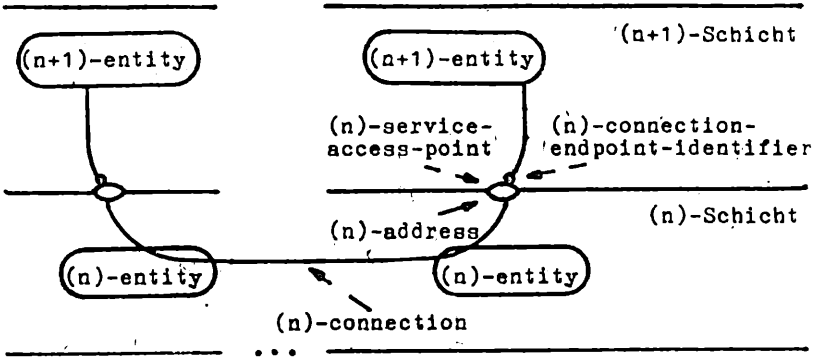



Abb. 8: Zusammenwirken von entities im OSI-Modell

Die (n+1)-entities nehmen ueber (n)-service-access-points den Dienst der (n)-Schicht in Anspruch. Ein (n+1)-entity wird seinem entsprechenden (n)-entity durch die (n)-address zugeordnet. Zur Abwicklung der Kommunikation zwischen den (n+1)-entities wird durch die (n)-Schicht eine (n)-connection aufgebaut. Der lokale Identifikator, mit dem ein (n+1)-entity die (n)-connection identifizieren kann, ist der (n)-connection-endpoint-identifier.

Zur Illustration ist der Ablauf einer Uebertragung in Abb. 9 dargestellt.

Die Nachrichten zur Realisierung des Protokolls werden logisch jeweils ueber eine Verbindung (connection) ausgetauscht (Rahmenpfeile).

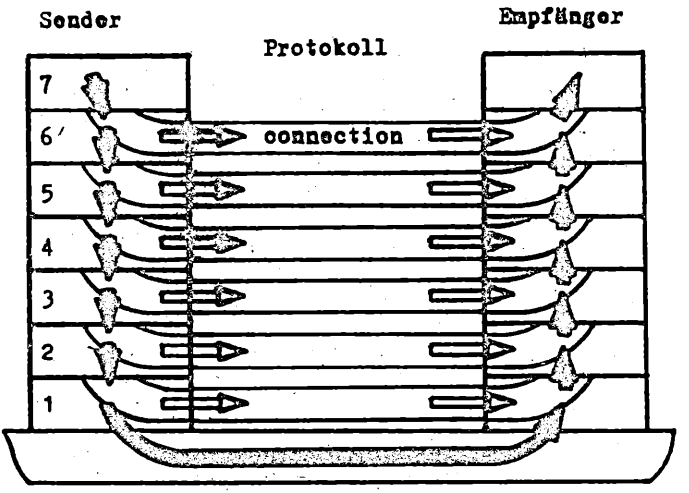



Abb. 9: Ablauf einer Nachrichtenuebertragung

Der reale Transport der Informationen in Form von Bitfolgen vollzieht sich dagegen im Sendersystem ueber alle 7 Schichten bis zum physischen Uebertragungsmedium, wird zum Empfaengersystem uebertragen und dort von Schicht zu Schicht nach oben gegeben (volle Pfeile). Dabei wird in der jeweiligen Schicht die logische Uebertragung ueber die Verbindung abgeschlossen.

Die oberste Schicht ist die Anwendungsschicht. Sie besteht aus den kooperierenden Anwendungskomponenten. Die unteren Schichten stellen stufenweise Kommunikationsdienste einer jeweils hoeheren Qualitaet bereit, durch die die Anwendungskomponenten ihre Kooperation abwickeln.
Die Schnittstelle zwischen zwei Schichten entspricht dem Qualitaetsuebergang und ist in OSI-Dienststandards (Service Definition) definiert. Die Arbeit innerhalb der Schichten wird durch OSI-Protokollstandards (Protocol Specification) geregelt.

Die einzelnen Schichten lassen sich folgendermassen kurz charakterisieren:

| | |
|---|---|
| Application | - Die Anwendungsschicht dient als Tor zwischen kommunizierenden Nutzern, durch das der Austausch von Informationen erfolgt. |
| Presentation | - Die Darstellungsschicht stellt die Informationen gegenueber den kommunizierenden Anwendungskomponenten so dar, dass die Bedeutung erhalten bleibt, waehrend syntaktische Unterschiede ausgeglichen werden. |
| Session | - Die Sitzungsschicht liefert die fuer die kooperierenden Darstellungskomponenten erforderlichen Mittel zur Organisation und Synchronisation ihrer Zusammenarbeit und zur Durchfuehrung des Datenaustauschs. |
| Transport | - Die Transportschicht gewaehrleistet eine transparente Datenuebertragung zwischen Sitzungskomponenten und sichert, dass diese zuverlaessig und kosteneffektiv ablaeuft. |
| Network | - Die Netzschicht stellt Mittel zum Aufbau, Betrieb und Abbau von Netzverbindungen zwischen OSI-Systemen bereit. |
| Data Link | - Die Verbindungsschicht fuehrt die Aktivierung, den Betrieb und die Deaktivierung von Datenverbindungen durch und entdeckt und korrigiert, wenn moeglich, Fehler der physischen Schicht. |

Physical - Die physische Schicht enthaelt die mechanischen, elektrischen, funktionellen und prozedurellen Mittel fuer die Aktivierung, den Betrieb und die Deaktivierung von physischen Verbindungen zur Bit-Uebertragung zwischen Verbindungskomponenten.

## 4. Rechnernetzdienste und Anwendungsgebiete im Rechnernetz DELTA

Das Rechnernetz fuer Forschung und Lehre DELTA /MEI81/, dass im Zentrum fuer Rechentechnik der AdW der DDR und im Rechenzentrum der TU Dresden entwickelt wurde, entspricht in seiner Systemarchitektur (Abb. 10) dem OSI-Modell zur Verkopplung offener Systeme.

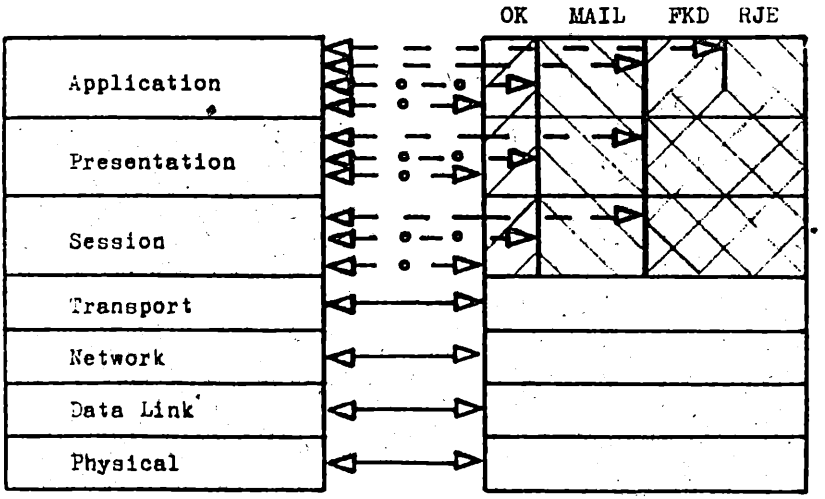



Abb. 10: Architekturmodell des Rechnernetzes DELTA

Die unteren vier Kommunikationsdienste sind einheitlich fuer alle Anwendungen realisiert. Ihre Protokolle wurden unter Beruecksichtigung der internationalen Diskussionen selbst entwickelt. Entsprechend der Verabschiedung der Standardprotokollempfehlungen der CCITT und ISO (z.B. /X.25, ISO861/) erfolgt die schrittweise Einfuehrung der neuen Protokolle.

Die oberen drei Schichten sind fuer jede der Anwendungen spezifisch entwickelt worden. Damit konnten der Umfang der entsprechenden Dienste auf die Erfordernisse der jeweiligen Anwendung beschraenkt werden und die bei der aufeinanderfolgenden Entwicklung gewonnenen Erkenntnisse sofort genutzt werden. Darueber hinaus wurde damit auf einfache Weise gewaehrleistet, dass nur die Softwarekomponenten der jeweils aktiven Anwendungen im Hauptspeicher der Rechnersysteme geladen sind.
Das bedeutete aber auch, dass fuer jede Anwendung fuer alle drei Schichten jeweils spezifische Protokolle entwickelt werden mussten.

Das Rechnernetz DELTA besteht aus dem Paketvermittlungsnetz KOMET /HAM80/ an das Verarbeitungsrechner vom Typ BESM6 und ESER angeschlossen sind und einem Terminalnetz ueber das mit KRS4201-Stapelterminals zu den Verarbeitungsrechnern zugegriffen werden kann (Abb. 11).

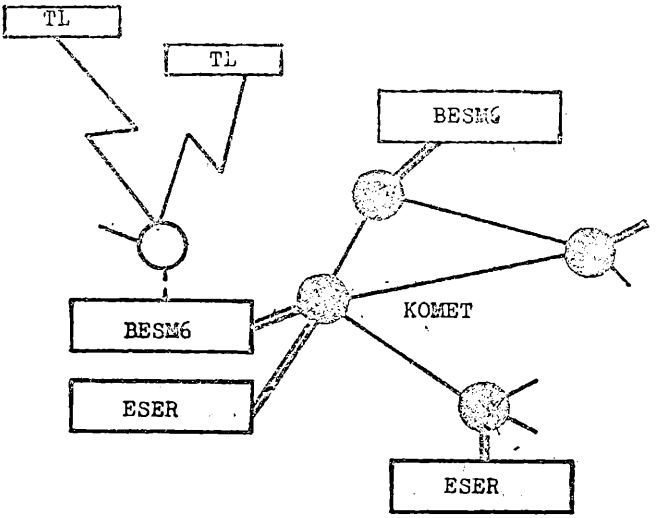



Abb. 11: Komponenten des Rechnernetzes DELTA

Im Rechnernetz DELTA werden z. Z. vier Nutzerdienste bereitgestellt, mit denen die gegenwaertigen Anforderungen der Nutzer aus der AdW der DDR, der AdL der DDR und dem Hochschulwesen befriedigt werden koennen.

Der Filekopierdienst ermoeglicht die Uebertragung von Files zwischen den Verarbeitungsrechnern. Die ESER-Rechner und BESM-Rechner untereinander koennen Files beliebiger Struktur austauschen. Die Uebertragung zwischen ESER- und BESM-Rechnern ist auf sequentielle Files beschraenkt.

Die Entfernte Auftragsbearbeitung realisiert die Verarbeitung von Stapeljobs an einem von der Ein-/Ausgabe unterschiedlichen Ort. Speziell kann fuer BESM-Jobs die Ein-/Ausgabe ueber Stapelterminals auf der Basis des Kleinrechners KRS4201 erfolgen.

Der Mailbox-Dienst, als Beispiel einer neuen Form der Telekommunikation, ermoeglicht das Formulieren von Briefen, deren Uebertragung zum Rechnersystem, auf dem sich die Mailbox

(Briefkasten) des Empfaengers befindet, und die Ablage in dieser Mailbox. Von dort kann der Empfaenger seine Briefe zu einem beliebigen Zeitpunkt abrufen.

Als vierter Dienst wird eine Gruppe von Einfachen Datenkommunikationsformen bereitgestellt, mit deren Hilfe innerhalb des Paketvermittlungsnetzes KOMET Telegramme uebermittelt und Lochstreifen uebertragen werden koennen, die Bediener der Verarbeitungsrechner ueber die Bedienkonsolen miteinander im Dialog kommunizieren koennen und die Nutzer an den Stapelterminals untereinander einen Dialog fuehren koennen.

Die gegenwaertigen Anwendungsgebiete dieser Dienste, entsprechend der in Pkt. 2 zitierten Eurodata'79-Studie, sind in Abb. 12 dargestellt.

| Dienst / Anwendung | PKD | RJE | MAIL | Einf.DK |
|---|---|---|---|---|
| Software-Entwickl. | X | X | X | X |
| wiss.techn.Berechn. | X | X | | |
| Zugriff zu DB | X | X | | |
| Mensch-Mensch-Komm. | | | X | X |
| Leitung/Verwaltung<br>Bankwesen<br>Buchungssysteme<br>Patientendaten<br>andere | | | | |

Abb. 12: Anwendungsgebiete der Nutzerdienste des Rechnernetzes DELTA

Sie entsprechen dem Charakter der wissenschaftlichen Einrichtungen, die ins Netz integriert sind. Die Dienste sind auch fuer andere Anwendungsgebiete, die sich beim Einsatz der Rechnernetztechnologie in anderen Bereichen der Volkswirtschaft ergeben, geeignet.

Literatur

/CHI81/ Childs, G. H. L.
The PRESTEL Viewdata System
COMNET'81 Working Papers
Statistical Publ. House, Budapest 1981

/EUR79/ Logica Ltd
Eurodata'79 - a Study of Data Communications in Western Europe, 1979-1987,
1980

/HAM80/ Hammer, D.
Kommunikationsnetze unter Nutzung der Paketvermittlung
ZfR-Informationen ZfR-I-80.03

/ISO861/ Draft Connection-oriented Basic Transport Protocol Specification
ISO/TC97/SC16 N 861

/ISO7498/ Data Processing - Open Systems Interconnection - Basis Reference Model
Draft Proposal ISO/DP 7498 (ISO/TC97/SC16 N 719)

/KIK82/ Кикутс,Я.А., Подвысоцкий,Ю.С., Ротанов,С.В., Кайя,Х.Х., Брикман,Г.А.
Протокол управления сеансом
препринт ИЭВТ, Рига 1982

/MCWA77/ McQuillan, J. M., Walden, D. C.
The ARPA Network Design Decisions
Computer Networks, Vol. 1 N. 5, 1977

/MED80/ Medcraft, W. F.
Data Network Planes for the UK
Proc. 5th ICCC, Atlanta
North-Holland Publ. Comp., 1980

/MEI81/ Meier, H. W., Heymer, V., u. a.
The Architecture of the Communication Services Hierarchy of the Computer Network DELTA and Experiences in its Implementation
COMNET'81 Working Papers
Statistical Publ. House, Budapest 1981

/MJA78/ Мямлин, А. Н.
Экспериментальная сеть ЭВМ коллективного пользования СЕКОП
препринт ИПМ АН СССР N. 30 , Москва 1978

/NERA82/ Немет,П., Разга,Т.
Некоторые сетевые и пользовательские вопросы создания услуг общего пользования на базе сети коммутации пакетов
Автоматика и вычислительная техника, N. 3, 1982

/POU73/ Pouzin, L.
Presentation and Major Design Aspects of the CYCLADES Computer Network
Proc. 3rd ASM-IEEE Communication Symposium, Tampa, Fl., 1973

/RAJ81/ Rajki, P. , u. a.
VIDEOTON Network System General Overview
COMNET'81 Working Papers
Statistical Publ. House, Budapest 1981

/X.25/ CCITT Recommendation "Interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks"
Rec. X.25

/YAK78/ Якубайтис, Э. А.
Транспортная сеть
препринт ИЭВТ, Рига 1978

Verfasser:
Dr. Volker Heymer
Akademie der Wissenschaften der DDR
Zentrum fuer Rechentechnik
1199 Berlin
Rudower Chaussee 5

Schreiter, Guenter

# Modulare Struktur eines Steuerprogrammes fuer einen freiprogrammierbaren Steuerrechner

## 1. Einleitung

Aufgrund sehr schnell wachsender Anforderungen und neuer Anwendungen haben sich freiprogrammierbare Steuerrechner in Datenuebertragungsnetzen international schnell durchgesetzt. Die freie Programmierbarkeit bietet folgende wesentliche Vorteile:

- Leichte Anpassung des DV-Systems an neue Geraetetypen oder Uebertragungsprozeduren ohne Aenderung der Geraetetechnik,
- Entlastung des Verarbeitungsrechners hinsichtlich Verarbeitungszeit und Speicherkapazitaet sowie
- guenstige Erweiterung des DV-Systems bei neuen Anwendungen, z.B. Rechnernetze.

## 2. Einordnung von Steuerrechnern in das E/A-System eines Verarbeitungsrechners

Die Ein- und Ausgabe von Daten ist stets Bestandteil des E/A-Systems einer EDVA. Das E/A-System der Datenfernverarbeitung (DFV) von ESER-EDVA ist durch selbstaendige Funktionseinheiten in mehreren Ebenen und mit zentraler Steuerung realisiert. Es besteht aus den Ebenen der

- Zentraleinheiten (ZE),
- Kanalsteuereinheiten (KSE),
- Geraetesteuereinheiten (GSE) fuer DFV und
- Datenendeinrichtungen (DEE).

Der Begriff Datenendeinrichtung (im ESER: Abonnentenpunkte) ist sehr weit gefasst. Es kann sich dabei sowohl um ein einfaches E/A-Geraet als auch um eine Rechenanlage handeln. DEE koennen entweder direkt oder indirekt ueber ein Datenuebertragungssystem an die GSE angeschlossen werden. Zwischen GSE und DEE koennen auch weitere Funktionseinheiten zwischengeschaltet sein.
Der Kleinrechner R 4201 kann bei entsprechender Konfiguration als Steuerrechner parallel zusammenarbeiten mit

- einer lokal angeschlossenen ESER-EDVA und mehreren entfernt angeschlossenen Verarbeitungsrechnern (VR) sowie
- mehreren lokal und entfernt angeschlossenen DEE.

Die Summe aller anschliessbaren Leitungen betraegt 15 und ist durch die max. Anzahl der Anschlusssteuereinheiten (AS) und die Nutzung der Echtzeituhr als Zeitgeber begrenzt. Sollen mehr als 15 DEE bedient werden, sind Leitungskonzentratoren erforderlich.
Der lokale Anschluss an die ESER-EDVA kann nur ueber den Multiplexkanal erfolgen. Ueber max. 12 Subkanaele lassen sich 12 Leitungen bedienen.
Beim Anschluss an einen entfernt aufgestellten VR sind Absprachen erforderlich ueber die

- physikalische Schnittstelle (funktionelle, elektrische und mechanische Eigenschaften) und
- logische Schnittstelle (Datenuebertragungsprozedur).

Der R 4201 ist somit geeignet fuer unterschiedliche Einsatzfaelle in der DFV (Bild 1).

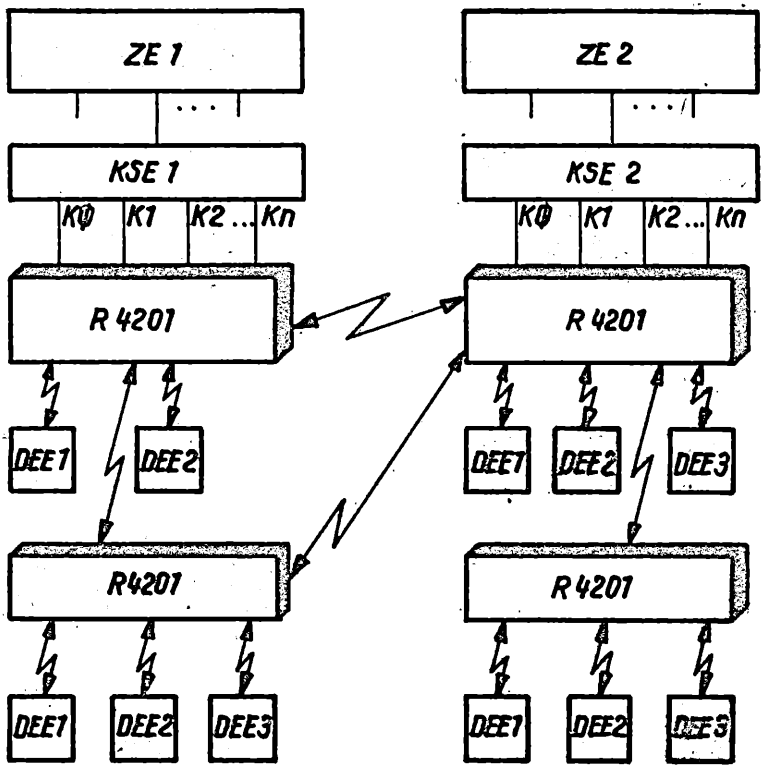



*Bild 1: Einsatzmöglichkeiten des R4201 im ESER, Reihe 1*

- lokal oder entfernt aufgestelltes Multiplexsteuergeraet fuer DFV (MPD),
- Leitungskonzentrator,
- Vermittlungsrechner,
- Abonnentenpunkt und
- Terminalsteuerrechner.

Trotz unterschiedlicher Einsatzmoeglichkeiten im E/A-System sind aus der Sicht der Steuerung die Aufgaben des Rechners die gleichen:

- lokaler und entfernter Anschluss an den Verarbeitungsrechner,

- Steuerung von DEE ueber verschiedene Typen von Uebertragungsleitungen mit unterschiedlichen physikalischen Schnittstellen und Geschwindigkeiten,
- Kode- und Schnittstellenwandlung,
- Zwischenpufferung der Daten,
- Fehlererkennung auf den Uebertragungsleitungen und Einleitung von Massnahmen zur Fehlerbehandlung.

Darueberhinaus besteht die Moeglichkeit der effektiveren Nutzung des VR, z.B. durch

- Zusammenarbeit mit mehreren Verarbeitungsrechnern und mit unterschiedlichen Rechnerarchitekturen,
- lokale und entfernte Dateein- und ausgabe,
- Realisierung kundenspezifischer Anforderungen,
- Arbeit mit mehreren Zugriffsmethoden, einschliesslich denen der lokalen Peripherie,
- umfassende Fehlererkennung- und korrektur,
- selbstaendigen Auf- und Abbau physikalischer und logischer Datenverbindungen,
- Moeglichkeiten der Datenvorbereitung- und pruefung,
- schnellere Umruestung auf geaenderte Netzwerkfunktionen, u.a.

Der Steuerrechner arbeitet nur mit den unmittelbaren Funktionseinheiten in der vertikalen bzw. horizontalen Ebene des E/A-Systems zusammen. Sie werden als E/A-Einheiten des Steuerrechners bezeichnet. E/A-Einheiten koennen sein:

- Subkanaele,
- entfernt aufgestellte GSE,
- E/A-Geraete,
- Leitungskonzentratoren,
- Abonnentenpunkte usw.

Fuer das Herstellen einer logischen Datenverbindung werden z.Zt. zwei Betriebsweisen genutzt:

- Aufrufbetrieb und
- Anforderungsbetrieb

Beim Aufrufbetrieb traegt eine E/A-Einheit die Gesamtverantwortung fuer die Datenuebertragung und legt fest, welche E/A-Einheit senden darf. Sie wird als Steuerstation bezeichnet. Es ist immer diejenige E/A-Einheit, an welcher der VR angeschlossen ist. Alle E/A-Einheiten, die keine Steuerstation sind, werden als Nebenstation bezeichnet. Eine Station die senden kann, heisst Hauptstation. Die Datenverbindung befindet sich im Betriebszustand "Unterordnungsbetrieb". Er wird hauptsaechlich bei Zweipunkt- und Mehrpunktverbindungen angewandt.
Beim Anforderungsbetrieb wird das Herstellen der logischen Verbindung gleichfalls durch die Steuerstation eingeleitet. In diesem Fall kann sowohl der VR als auch die DEE die Steuerstation sein. In der vorliegenden Version wird diese Betriebsweise nicht unterstuetzt.

## 3. Steuerprogrammkomponenten

### 3.1. Begriffe

Der Prozess (Task) ist die kleinste in die Ablaufplanung eines Steuerprogrammes eingehende Einheit. Er beginnt mit einem Anfangszustand und endet mit einem Endzustand. Ein Prozess besteht aus einer oder mehreren Aktionen, die mit einem Anfangsereignis beginnen und mit einem Endeereignis enden.
Die Prozesse, die auf dem Steuerrechner ablaufen, kann man unterteilen in:

- Systemprozesse und
- E/A-Prozesse.

Systemprozesse werden von der Komponente "Ablaufsteuerung" durch Systemtasks verwaltet, E/A-Prozesse von der Komponente "E/A-Steuerung" durch E/A-Tasks.

## 3.2. Ablaufsteuerung

Die Komponente Ablaufsteuerung besteht aus folgenden Tasks:

- Behandlung des Ereignisses Sammelunterbrechung,
- Behandlung des Zeitereignisses,
- Behandlung des Ereignisses Spannungsfehler,
- Regieprogramm und
- Dynamischer Stop.

Das Regieprogramm ist fuer die Ablaufsteuerung die wichtigste Task. Sie verwaltet die parallel zu bearbeitenden E/A-Prozesse, die sich in einem der Zustaende "Wartend", "Bereit" oder "Aktiv" befinden, durch die Zustandsaenderungen "Starten","Zuordnen" oder "Beenden" (Bild 2).

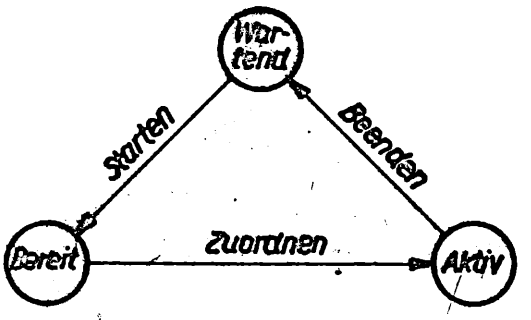



Bild 2: Zustandsdiagramm von E/A-Prozessen

Eine Zustandsaenderung ist als Aktion in einem System- oder E/A-Prozess anzuzeigen.

Das Regieprogramm wird nach Auswertung eines Ereignisses aufgerufen und startet nach Behandlung aller anstehenden Zustandsaenderungen die Task "Dynamischer Stop".
Die Task "Dynamischer Stop" wird dem Prozessor nach der Systemanlaufbehandlung zugeordnet. Sie wird verdraengt, sobald ein Ereignis eintrifft.
Die Tasks zur Behandlung des Ereignisses Sammelunterbrechung und des Zeitereignisses arbeiten kooperativ mit den E/A-Tasks zusammen.
Die Task zur Behandlung des Ereignisses Spannungsfehler fuehrt zum Halt des Prozessors. Nach anschliessendem (Re-) Start wird in die Task "Dynamischer Stop" verzweigt.

### 3.3. E/A-Steuerung

Aufgrund von abgrenzbaren Teilfunktionen sind folgende wesentliche E/A-Tasks zu nennen:

- E/A-Initiator,
- Sammelunterbrechungsbehandlung,
- Zeitfehlerbehandlung und
- E/A-Fehlerbehandlung.

Um eine E/A-Einheit bedienen zu koennen, ist ein ordnungsgemaesses Zusammenspiel aller E/A-Tasks erforderlich. Dieser Algorithmus wird Treiber genannt.
Die E/A-Task "Sammelunterbrechung" wird von der Systemtask zur Behandlung des Ereignisses Sammelunterbrechung aufgerufen, nachdem diese die AS ermittelt hat. Wurde eine Sammelunterbrechung durch einen E/A-Fehler verursacht, wird die E/A-Task "E/A-Fehlerbehandlung" aktiviert, ansonsten die E/A-Aktion des Treibers ausgefuehrt. In der Task ist anzuzeigen, ob der E/A-Prozess in den Zustand "Wartend" (Aktion "Beenden") zu ueberfuehren ist oder im Zustand "Aktiv" (Aktion leer) bleibt.

Die E/A-Task "E/A-Initiator" wird von der Systemtask "Ablaufsteuerung" aufgerufen und die E/A-Task "Zeitfehlerbehandlung" von der Systemtask zur Behandlung des Zeitereignisses.
Da ueber den gleichen Typ einer AS unterschiedliche E/A-Einheiten angeschlossen werden koennen, ist es zweckmaessig, die Steuerung der AS von der Steuerung der E/A-Einheiten zu trennen. Deshalb wird jede AS durch einen physikalischen Treiber und jede E/A-Einheit durch einen logischen Treiber bedient.
Das Zusammenwirken der einzelnen Hardware- und Softwarekomponenten zeigt Bild 3.

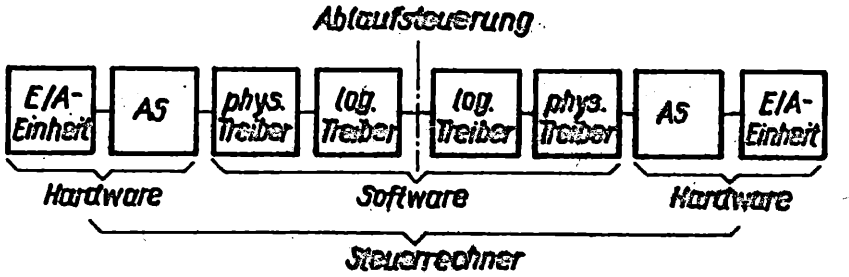



**Abb. 3: Zusammenwirken zwischen Hardware- und Softwarekomponenten im Steuerrechner**

Logische Treiber wurden bisher realisiert fuer:

- Multiplexkanal (AP64, Fernschreiber, IBM 2741 und ES 6012 fuer die Zusammenarbeit mit BTAM OS/ES),
- entfernt aufgestellte MPD auf der Basis der AP64-Prozedur (AP64-Emulator),
- AP64
- SM 4000/BD 4000,
- VT 340,
- PBT 4000 und
- CR 600.

Physikalische Treiber sind vorhanden fuer die AS1, AS2, AS4, AS8 und AS9.

Zur Steuerung einer AS gehoeren alle physikalischen Aufgaben wie Herstellen der E/A-Bereitschaft, Ein- und Ausgabe eines Datenbytes, Behandlung von Statusmeldungen.
In der Steuerung einer E/A-Einheit sind alle logischen Aufgaben zusammengefasst, wie Starten und Beenden einer E/A-Operation, Ausblenden und Einblenden von Uebertragungssteuerzeichen, Kodierung, Pufferung usw.
Alle zur Steuerung einer AS benoetigten statischen und dynamischen Informationen werden in einem Statussteuerblock (SCB) und die einer E/A-Einheit in einem Einheitensteuerblock (UCB) gespeichert.
Typische statische Informationen eines SCB sind:

- Maskenbit,
- Adresse der AS,
- Statusregister,
- Time-out-Zaehler usw.

Zu den statischen Informationen eines UCB gehoeren:

- Adresse des SCB,
- Adresse der Tabelle aller Eintrittspunkte des Treibers
- Adresse der Datenuebertragungsprozedur (DUEP) usw.

Die in der DDR realisierten Implementierungen von Steuerprogrammen gehen davon aus, dass nur DEE unterstuetzt werden koennen, fuer die eine DUEP vorhanden ist. Die DUEP wird von der Zugriffsmethode (ZGM) fuer DFV im VR zusammengestellt und die zugeordnete Uebertragungszeichenfolge nach dem Durchschalteprinzip uebertragen. Das ist bspw. beim MPD1A-Emulator der Fall. Zur Unterstuetzung von DEE, die nicht zum Modellbestand des ESER gehoeren, zum Anschluss mehrerer VR u.a. Aufgaben ist eine Trennung von Nachrichtenkopf und Nachrichtentext erforderlich.

Dafuer ist eine Nachrichtenvermittlung mit blockweiser Datenuebertragung wesentlich besser geeignet. Das entsprechende Verfahren wird "Emulation zur prozedurfreien Datenuebertragung" bezeichnet und ist Aufgabe des Kanaltreibers. Einzelheiten dazu sind aus /3/ zu entnehmen.
Grundlage fuer eine Emulation ist stets die Nachbildung des vollstaendigen Algorithmus zur Informationsuebertragung von Kanalprogrammen. Kanalprogramme werden in der Ebene der ZE erzeugt. Fuer die GSE ist es belanglos, ob dafuer ueberhaupt ein Betriebssystem bzw. eine ZGM benutzt wird. Praktisch besteht jedoch eine Einschraenkung, da das Achtungsbit vom Statusbyte nicht zum Multiplexkanal uebertragen werden.

## 3.4. Auftragssteuerung

Die Auftraege, die das Steuerprogramm zu verwalten hat, sind E/A-Auftraege. Ueber die AS4 koennen je Subkanal ein E/A-Auftrag und ueber jede AS2 bzw. AS8 je ein E/A-Auftrag parallel abgewickelt werden. Somit ist die Anzahl der zu verwaltenden E/A-Auftraege vor der Laufzeit des Steuerprogramms bekannt und kann definiert werden.
Jeder E/A-Auftrag wird in einem Tasksteuerblock (TCB) beschrieben und enthaelt statische und dynamische Informationen:

- Verweis zur Steuerstation,
- Verweis zur Nebenstation
- Laenge des E/A-Puffers
- Adresse des E/A-Puffers und
- Verweis zur Liste der Steuerrechner

Jeder E/A-Auftrag besteht aus zwei E/A-Operationen. Eine Datenuebertragung vom VR zur DEE erfordert eine Eingabeoperation

vom VR und anschliessend eine Ausgabeoperation zur DEE.
Die Auftragssteuerung hat also nach Beendigung einer E/A-Operation fuer die Umkehr der Uebertragungsrichtung zu sorgen.
Bei Zweipunktverbindungen ist die Zuordnung zwischen ankommender und abgehender Verbindung vor der Laufzeit des Steuerprogramms bekannt. Eine Adressierung entfaellt. Bei Mehrpunktverbindung ist eine von mehreren abgehenden Verbindungen bei nur einer ankommenden Verbindung auszuwaehlen. Es ist ein Adressverweis zur Auswahl der Verbindung erforderlich, der bspw. in der DUEP enthalten sein kann.
Erfolgt der Zugriff vom VR zu einer DEE ueber mehrere Steuerrechner muessen die Adressen der Steuerrechner eindeutig im Netz und die Adressen der am Steuerrechner angeschlossenen DEE eindeutig vergeben sein. Damit ist es moeglich, dass jeder VR im Netz jede DEE eines Steuerrechners adressieren kann.
Ein Steuerrechner muss unterscheiden, ob er einen weiteren Steuerrechner oder eine DEE adressieren soll. Dazu muessen ihm die Adressen aller Steuerrechner im Netz sowie die Adressen aller angeschlossenen DEE bekannt sein.
Eine Netzadresse wird ueber die im Bild 4 dargestellte Listenstruktur interpretiert. Wird ein Steuerrechner adressiert, enthaelt das Element in der Liste der Steuerrechner einen Adressverweis zu einem Steuerblock. Die DEE-Adresse wird nicht weiter ausgewertet. Wird eine DEE adressiert, enthaelt das Element in der Liste der Steuerrechner einen Adressverweis zur Liste der DEE. Den Adressverweis enthaelt dann das entsprechende Element in der Liste der DEE.

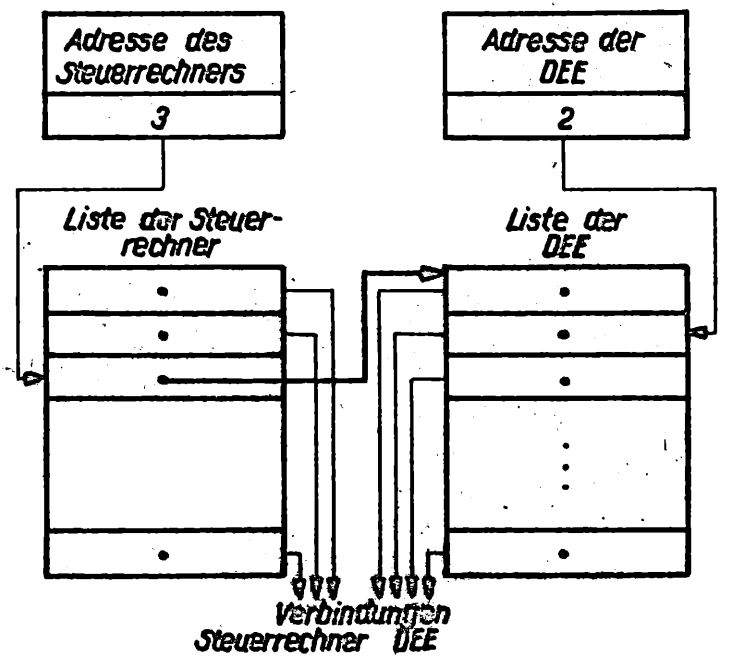



Bild 4: Adressenlnterpretation im Steuerrechner

Dieses Adressierungsprinzip wird bspw. in der Systemnetzwerkarchitektur (SNA) /1/ angewendet. Die VR werden als Hosts, die Steuerrechner als Subareas und die DEE als Komponenten bezeichnet.

Die AP64-Prozedur des ESER ist zur Auftragssteuerung gut geeignet. Sie enthaelt zum Aufbau der logischen Verbindung u.a. zwei Adressenbytes: eine Steuereinheit- und eine Komponentenadresse. Der Wertevorrat der Steuereinheitadresse betraegt 96 Zeichen (Wertebereich '20' bis '7F'); der Komponentenadresse 48 Zeichen (Wertebereich '30' bis '5F').

Damit laesst sich - ohne Aenderung der Geraetetechnik und Eingriffe in das Betriebssystem einer ESER-EDVA! - ein Rechnernetz mit 96 Subareas aufbauen.

## 3.5. Kernspeicherverwaltung

Aufgrund des Prinzips der Nachrichtenvermittlung spielt die Bereitstellung von E/A-Puffern zur Zwischenspeicherung von Daten eine besondere Rolle in der Verwaltung des Kernspeichers. Beim Aufrufbetrieb ist es ausreichens, je E/A-Auftrag einen E/A-Puffer zu definieren, der jeweils der Hauptstation zugeordnet wird.
Fuer Testzwecke ist die Rueckverfolgung von E/A-Operationen ein unentbehrliches Hilfsmittel. Alle physikalischen Treiber protokollieren jede Datenuebertragung ueber die AS. Eine Protokollierung besteht aus drei Eintraegen:

- Ereignis (Datenbyte, Statusbyte),
- Verweis zur ausgefuehrten Aktion,
- Adresse des Subkanals (nur AS4),

Das Protokoll wird in einem Umlaufpuffer gespeichert.

## 4. Schlussbetrachtung

Das vorgestellte Steuerprogramm fuer einen Steuerrechner auf der Basis des R 4201 hat sich bereits in der Praxis bewaehrt. Als wesentliche Vorteile gegenueber bisherigen Systemunterlagenentwicklungen werden gesehen:

1. Bedienung von DEE, die nicht zum Modellbestand der Abonnentenpunkte gehoeren, besonders beim Einsatz als MPD.
2. Aufbau einfacher Rechnernetze ohne Aenderung an der Geraetetechnik und bei minimaler Aenderung der Systemunterlagen.
3. Unterstuetzung mehrerer DEE beim Einsatz als Abonnentenpunkt
4. Einheitliches und dem Anwender zugaengliches Steuerprogramm fuer alle Einsatzmoeglichkeiten.
5. Guenstiger Offline-Test bzw. Fehlersuche.

Literaturverzeichnis

/1/ Kohl,H.: Systems Network Architecture: Rechnernetze. IBM-Nachrichten, 26(1976)233/234

/2/ Schnupp,P.: Rechnernetze: Walter de Gruyter, Berlin New York, 1978

/3/ Schreiter,G.: Emulator fuer prozedurfreie Datenuebertragung. Rechentechnik Datenverarbeitung 18(1981)9

/4/ Wettstein,H.: Aufbau und Struktur von Betriebssystemen. Carl Hanser Verlag, Muenchen Wien, 1978

/5/ ISO/TC97/SC16, Open System Interconnection, 1978

Verfasser:

Dipl.-Ing. Guenter Schreiter
IH Dresden
Sektion Informationsverarbeitung
Rechenzentrum
8019 Dresden
Hans-Grundig-Str. 25

Sacklowski, Ulrich

# Gedanken zu Gestaltungsaspekten der Bildschirm-Interaktionsschnittstelle und ihren programmiertechnologischen Realisierungen

Gestaltungsaspekte der Bildschirm-Interaktionsschnittstelle (BS-IAS), die mittels Programme realisiert werden können, sind solche, die mit
- Bildinhalt und Bildform,
- Bildfolge und
- Zeitüberlegungen

zusammenhängen.

Die folgenden Ausführungen beschränken sich auf alpha-numerische Bildschirme. Hinsichtlich der im Dialog zu behandelnden Aufgaben ist an solche
- des Programmierens,
- der rechnergestützten Unterweisung,
- an die mittels Auskunfts- und Informationssysteme zu lösenden,
- sowie an darüber hinausgehende in der Büroautomation anfallende

gedacht.

Gestaltungsaspekte der BS-IAS werden häufig subsumiert unter den Begriffen Benutzerakzeptanz oder Benutzerfreundlichkeit. Untersetzende Teilaspekte werden in der Literatur zahlreich genannt und behandelt. Unbefriedigend, und eine umfassende programmiertechnologische Erschließung erschwerend, sind die unterschiedlichsten Herangehensweisen der Autoren und die daraus resultierende Begriffsvielfalt, entlehnt den verschiedensten Wissensgebieten. So stehen Überlegungen
- zur Ästhetik und einem zeitgenössischen Stil (Shneiderman 1979)

neben Betrachtungen
- zu fachspezifischen Dialogsprachen,
- zur Antwortzeitproblematik,

oder etwa dem
- Benutzertemperament (Sacklowski 1981).

Ein integraler und damit die verschiedensten Begriffe integrierender Ansatz scheint mit den Aussagen der Kommunikationswissenschaften, insbesondere der linguistischen Kommunikation, gegeben zu sein, wobei, da es sich um Arbeitstätigkeiten handelt, Erkenntnisse der Arbeits- und Ingenieurpsychologie einbezogen werden müssen.

Dehning und Maaß haben 1977, ausgehend von zwei Kommunikationsmodellen, Erwartungen abgeleitet, die ein Mensch an eine zwischenmenschliche Kommunikation knüpft, und die nach ihrer Meinung auch in der Mensch-Rechner-Interaktion ihre Relevanz haben. Ihre genannten Erwartungen sind folgende:

Ein Mensch erwartet,

1. daß sich ein gemeinsamer Kode finden läßt, und daß auch der Kode des Partners erweiterbar ist,
2. beim Partner ein Grundwissen und die Fähigkeit, neue Erfahrungen zu sammeln.

Er erwartet, daß der Partner

3. Rücksicht auf die Redekonstellation nimmt,
4. nach einem partnertaktischen Programm handelt,
5. von psychischen und physischen Gegebenheiten abhängig ist,
6. eine Präzisionsregelung anwendet,
7. eine zeitliche Regelung anwendet.

Er erwartet für sich selbst,

8. daß er das Verhalten des Partners beeinflussen kann,
9. Möglichkeiten zur Präzisionsregelung,
10. Möglichkeiten zur zeitlichen Regelung,
11. eine Reaktion des Partners bezüglich seines eigenen Verhaltens.

Die genannten Autoren transformierten diese Erwartungen auf die Mensch-Rechner-Interaktion und ordneten ihnen in der Literatur genannte BS-IAS-Gestaltungsmerkmale zu, wie sie für sogenannte „naive" Benutzer bei Routine- und bei neuartigen Arbeiten von Bedeutung sind.

Für eine allgemeine programmiertechnologische Erschließung dieses Ansatzes wäre seine tiefergehende Qualifizierung erforder-

lich. Dies würde bedeuten:

1. Bei der Differenzierung der erwartungs-realisierenden Merkmale eine generelle Unterlegung der Hauptaxiome der Kommunikation (z.B. mittels der Axiome nach Baacke (1973)).
2. Eine geordnete und klare Nennung der Merkmale, insbesondere in Hinblick auf ihre mögliche Formalisierung, schrittweise Verfeinerung und Abstraktion, und
3. eine stärkere Einbeziehung der Erkenntnisse der optischen Kommunikation und damit der Gestaltgesetze der Psychologie.

Ein wesentliches Ergebnis solcher Untersuchungen sind Qualitätsmerkmale des BS-IAS.

Qualitätsmerkmale für Programme und Datenverarbeitungsprojekte werden seit langem diskutiert. Sie erscheinen jedoch für die BS-IAS als noch zu mächtig, noch nicht spezifisch genug, als daß sie beispielsweise in der Spezifizierungsphase als ausreichende Richtlinien gelten könnten.

Ausgehend vom kommunikativen Ansatz erarbeitete Qualitätsmerkmale sind hinsichtlich der Aufgabe, des Benutzers und des Rechnersystems quantitativ untersetzbar (Bild 1).

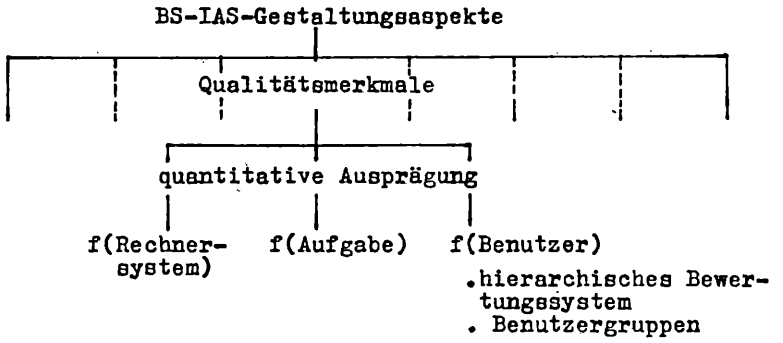



Bild 1: Qualitative und quantitative Ausprägung von BS-IAS-Gestaltungsaspekten

Einige Bemerkungen zur aufgaben- und benutzer-abhängigen quantitativen Ausprägung:

1. zur aufgaben-abhängigen, - hier speziell anhand der Präzisionsregelung. Ein Bestandteil der Präzisionsregelung ist,

daß bei fehlerhaftem Dialogverlauf die Möglichkeit des Rücksprungs mit erneutem Wiederanlauf geboten wird.
Als Beispiel für nur eine Rücksprungmöglichkeit, nämlich zum Dialoganfang, seien die öffentlichen Dialog-Fahrkartenautomaten mit Bildschirm genannt, für zahlreiche Rücksprungmöglichkeiten Dialoganwendungen, die der Erstellung von Steuerkarten für ein größeres Programm dienen.

2. zur benutzer-abhängigen, - hier nur zur näheren Erläuterung der beiden angeführten Unterpunkte.
   (a) Hierarchisches Bewertungssystem: Gemeint ist das hierarchische Bewertungssystem von Arbeitstätigkeiten der Arbeits- und Ingenieurpsychologie (Hacker 1980). Es stuft Arbeitstätigkeiten von ausführbar, über schädigungslos, beeinträchtigungslos bis persönlichkeitsfördernd ein.
   (b) Benutzergruppen: Gemeint sind gruppenspezifische Merkmalsausprägungen, die sich z.B. aus
   - den Erfahrungen und EDV-Kenntnissen des Benutzers,
   - der Benutzungshäufigkeit der Dialoganwendung
   - und etwa den Alters- und Temperamentscharakteristika ergeben.

Die nun folgenden Ausführungen hinsichtlich der programmiertechnologischen Realisierung der Gestaltungsaspekte reduzieren sich auf die Programmierphasen der Spezifizierung und des Entwurfs.

In den zurückliegenden Jahren haben sich die Bemühungen verstärkt, die Spezifizierungsphase programmiertechnologisch zu unterstützen. Es sind Methoden, Sprachen und Werkzeuge entwickelt worden, die den Spezifizierungsprozeß und die Spezifikation einschließlich deren Analyse und damit der Überprüfung ihrer inneren Konsistenz, unterstützen. Zwei Hauptvertreter hierbei sind mit den Systemen SADT und ISDOS gegeben, letzteres mit seinem Sprachteil PSL und seinem Analyseteil PSA. SADT und ISDOS unterscheiden sich konzeptionell in ihrer Herangehensweise. Während ersteres einen integralen Ansatz verfolgt, entsprechend einer TOP-DOWN-Strategie, folgt PSL von ISDOS einem singulären Ansatz und damit einer BOTTOM-UP-Strategie. Obwohl die sprachlichen Mittel beider Systeme teilweise sehr mächtig sind, fehlen jedoch

insgesamt Beschreibungsmittel für den Mensch-Rechner-Dialog.

Die Literatur nennt andere Ansätze für diesen Anwendungsbereich, jedoch sind sie nicht so umfassend.

Bei ihnen können zwei Herangehensweisen unterschieden werden
- die prozeßorientierte und
- die zustandsorientierte.

Beim prozeßorientierten Ansatz werden beim Entwurf des Dialogsystems die Teilaktivitäten über parallele, rivalisierende Prozesse realisiert. Ein solcher Systementwurf gewinnt gegenüber einem zustandsorientierten besonders dann an Übersicht, wenn das Dialogsystem komplex ist, - viele Dialogzustände hat. Als Beispiel dafür seien eine weitreichende Kodetoleranz, mit Formatfreiheit und Freiwortanalyse, und adaptive Systemlösungen genannt.

Weiter verbreitet als der prozeßorientierte Ansatz ist der zustandsorientierte. Ein Hauptgrund wird in seiner klaren Darstellung des Dialogflusses liegen.

Der zustandsorientierte Ansatz geht auf die Theorie endlicher Automaten zurück und beschreibt für jeden Dialogzustand die möglichen Eingaben und die daraus resultierenden Folgezustände. Von den zu seiner Darstellung entwickelten Hilfsmitteln soll kurz auf die Petrinetze, die petrinetzähnlichen Instanzennetze, Interaktionsdiagramme und Dialogablauftabellen eingegangen werden.

1. Zu Petrinetzen und petrinetzähnlichen Instanzennetzen:
   Das Beispiel in Bild 2 zeigt einen Ausschnitt eines Dialogsystems zur Programmentwicklung. Je nachdem, ob im Petrinetz die Bedingung „Compilation" oder „Keine Compilation" markiert ist, findet das Ereignis E1 oder E2 statt.

   Petrinetze eignen sich für den Nachweis der Sicherheit und Konfliktfreiheit der Programme von Dialogsystemen. Ihr Beschreibungsniveau ist jedoch sehr niedrig, die Übersichtlichkeit geht schnell verloren. Deshalb wurden Petrinetze erweitert und dabei neben der Darstellung des Steuerflusses besonderes Gewicht auf die Wiedergabe des Datenflusses gelegt (Bild 3).

Die hiermit gewonnene Transparenz geht allerdings auf Kosten von Einzelinformationen. Diese können dann beispielsweise mittels Pseudocode hinzugefügt werden.

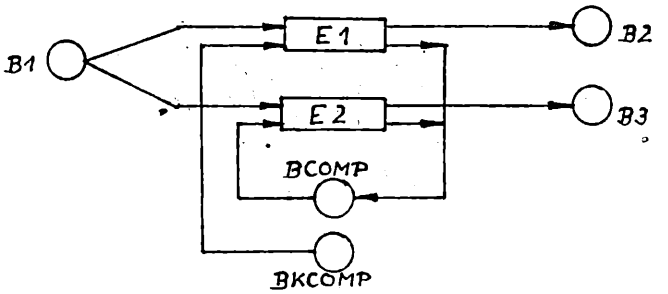
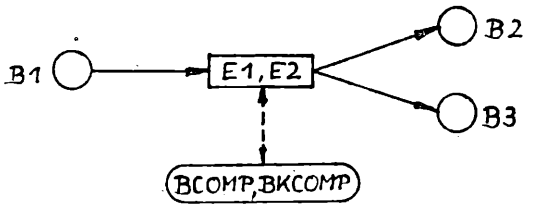



Bild 2: Beispiel für ein Petrinetz (Schmid 1973)




```
M:   if   B1 = marked
     then   if BCOMP,BKCOMP = BCOMP
            then mark (B3)
            else mark (B2)
                 BCOMP,BKCOMP:= BCOMP
            fi
     else   goto M
     fi
```

Bild 3: Beispiel für ein petrinetzähnliches Instanzennetz (Schmid 1973). Dabei verwendete Kanalklassen:

(1) Kanal mit der gleichen Funktion wie Bedingungen in Petrinetzen

Kanal: (B1) ; Kante: ——►

(2) Kanal durch Zusammenfassen von ≧ 2 Bedingungen

Kanal: (B1,B2) ; Kante: — — —►

(3) Kanal mit komplexen Informationen (z.B. Programmbaustein)

Kanal: (PBS) ; Kante: — — —►

2. Zu Interaktionsdiagrammen:
   Interaktionsdiagramme sind eine spezielle Art der Zustandsdiagramme, die sich besonders für die Spezifizierung und den Entwurf von Dialogsystemen nach dem Verfahren der schrittweisen Verfeinerung eignen. Ihre Syntax und Semantik sollen anhand eines Beispiels angedeutet werden. Es entstammt dem medizinischen Bereich und ist Untersuchungen von Budde u.a. (1980) entnommen (Bild 4).
   Zur Erläuterung von Bild 4:
   - Dreiecke bedeuten Anfangs- bzw. Endzustand
   - Kreise entsprechen Interaktionspunkten; mit eingefügtem "T" = Terminal
   - einfach gerahmte Rechtecke entsprechen einfachen Zuständen, sie beinhalten Verarbeitungsabläufe ohne weitere Interaktionen mit dem Dialogpartner
   - doppelt gerahmte Rechtecke entsprechen "komplexen Zuständen", sie werden durch zusätzliche Interaktionsdiagramme verfeinert und in ihnen können weitere Interaktionen mit dem Dialogpartner vorkommen
   - Striche mit darangeschriebenen Bedingungen dienen der Darstellung von Zustandsübergängen.

   Der untere Ablauf des Beispiels gibt eine erste Verfeinerung des komplexen Zustandes "Rezeptdruck" wider.
   Ein solcher Dialogablauf in graphischer Form ist auch in Tabellenform möglich. Diese wurde bei uns vom VEB Carl Zeiss Jena publiziert (Rothardt 1979).

Abschließend zwei kritische Bemerkungen zu allen genannten Darstellungsformen:

1. Allen fehlen Zeitangaben. Auf Grund der erkannten hohen Relevanz der Antwort-Zeit-Problematik und der einfachen Einfügungsmöglichkeit sollten Antwortzeitforderungen mit aufgenommen werden.

   Im Beispiel von Bild 4 sind die maximalen Antwortzeiten in Sekunden in eckigen Klammern hinzugesetzt worden.

2. Alle lassen das "GOTO" und damit einen nichthierarchischen Steuerfluß zu. Dies ist sehr bedenklich, da Erkenntnisse der Arbeits- und Ingenieurpsychologie hinsichtlich der geistigen

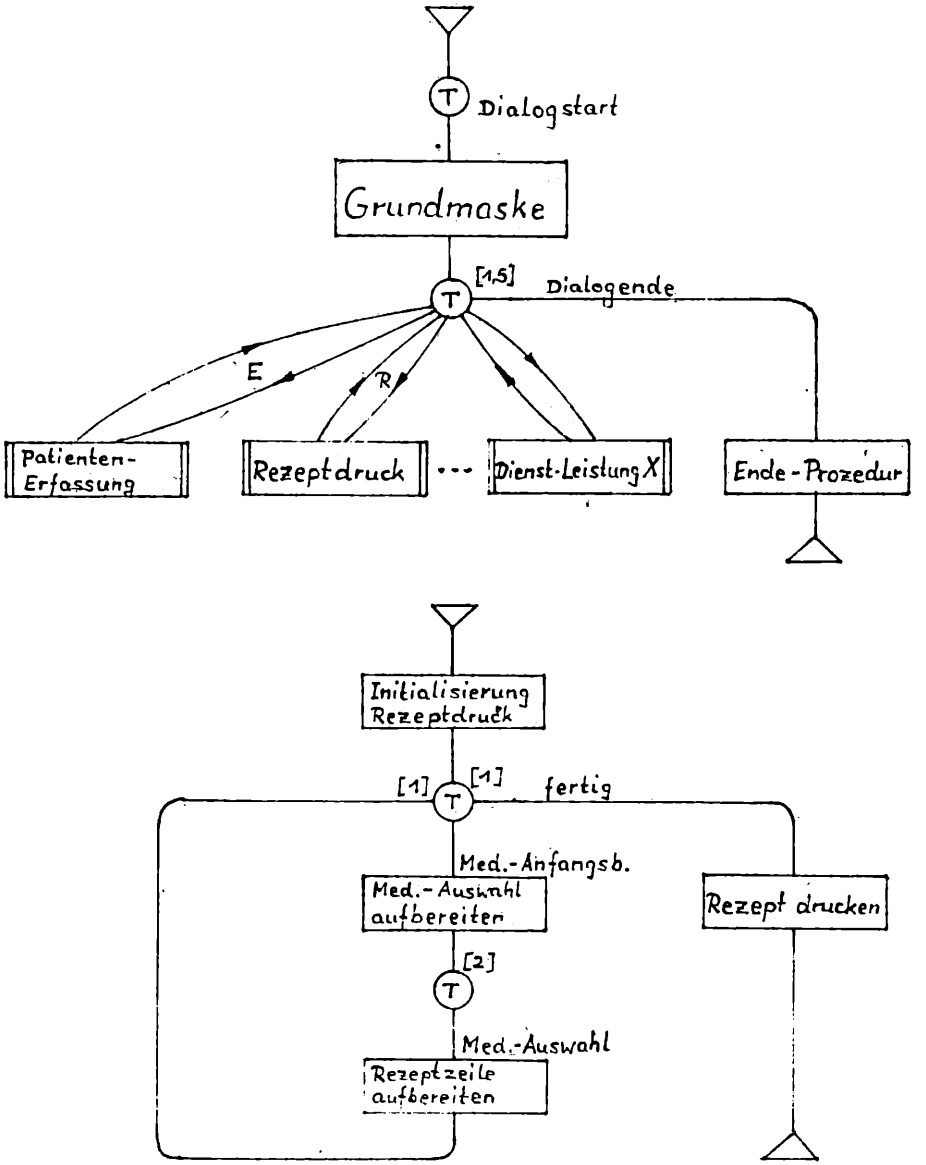



Bild 4: Beispiel für ein Interaktionsdiagramm mit Verfeinerung des „komplexen Zustandes" Rezeptdruck

Repräsentation von Arbeitstätigkeiten, also auch bei der Arbeit mit einem Dialogsystem, besagen, daß hierbei grundsätzlich von einer hierarchischen Struktur und Ablaufsteuerung auszugehen sei (Hacker 1980; Bild 5).

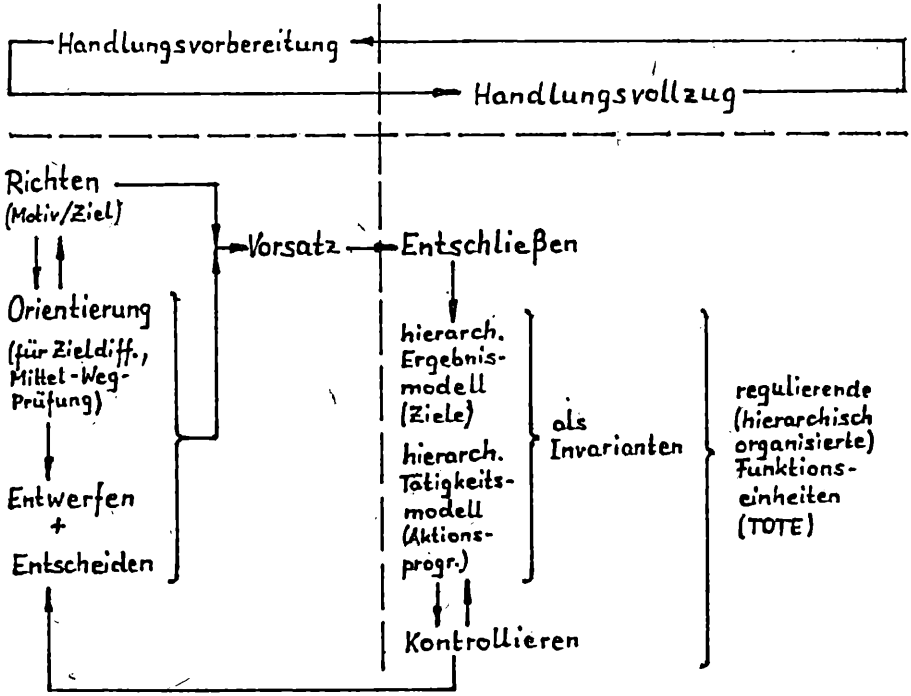



Bild 5: *Grob schematisierter Überblick über Glieder und Zusammenhänge der psychischen Regulation von Arbeitstätigkeiten*

Innerhalb des Schemas von Bild 5 ist der rechte Teil von Interesse, also der den Handlungsvollzug charakterisierende. Die dortigen Invarianten entsprechen relativ beständigen tätigkeitsregulierenden psychischen Gebilden und sind Teile des inneren Modells oder, wie man auch sagt, des operativen Abbildsystems. Zu ihrer Struktur sagt Hacker: „Vornahmen und ihrer Realisierung dienende Pläne stehen in einem durch die Logik der Aufgabe bedingten Unterordnungsverhältnis, sie sind hierarchisch 'verschachtelt'. Der Hierarchie der Zielsetzung entspricht eine Hierarchie der Pläne, allgemeiner der Aktionsprogramme".

Und an anderer Stelle schreibt Hacker: „Das operative Abbildsystem reguliert Arbeitstätigkeiten vermittels funktioneller Einheiten von in die Zukunft ausgreifenden, hierarchisch gestaffelten Vornahmen und zu deren Verwirklichung dienenden, entsprechend organisierten Handlungs- oder Aktionsprogrammen, die offenbar gleichfalls hierarchisch organisierte Rückkopplungsprozesse (Kontrollprozesse mittels TOTE-Einheiten) einschließen".
Dies bedeutet: Äquivalenz in Struktur und Steuerfluß zwischen
1. der im Dialog zu lösenden Aufgabe und
2. ihrer geistigen Repräsentation und
3. des entworfenen und implementierten Dialogsystems

läßt nur eine durchgängig hierarchische Struktur und einen durchgängig hierarchischen Steuerfluß zu.

Literaturverzeichnis

/1/ Baacke, D.: Kommunikation und Kompetenz, München 1973

/2/ Budde, R., Schnupp, P., Schwald, A.: Untersuchung über Maßnahmen zur Verbesserung der Software-Produktion (Teil 1), München Wien 1980

/3/ Dehning, W., Maaß, S.: Kommunikative Aspekte der Mensch-Computer-Interaktion, Universität Hamburg, Mitteilung Nr. 43 des Institutes für Informatik, 1977

/4/ Hacker, W.: Allgemeine Arbeits- und Ingenieurpsychologie, Berlin 1980

/5/ Rothardt, G.: Tabellenmethode zur Dokumentation des Mensch-Maschine-Dialoges, rd 10/1979

/6/ Sacklowski, U.: Individuell-psychologische Aspekte des EDV-Einsatzes, insbesondere der Mensch-Computer-Interaktion, und ihre Bewertung, Beitrag auf dem II. Kolloquium „LO und EDV" der Universität Rostock, 1981 (unveröffentlicht)

/7/ Schmid, H.A.: Zur Struktur von Dialogsystemen, Diss. TH Karlsruhe 1973

/8/ Shneiderman, B.: Human Factors Experiments in Designing Interactive Systems, Computer, December 1979

Verfasser:

U. Sacklowski

Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum

1086 Berlin

Unter den Linden 6

Detering, Rolf
Stantschev, Valentin

## Vergleich rechnerunterstützter Spezifikationsverfahren

### 1. Einführung

In den letzten Jahren entstand ein wachsendes Interesse an allen Phasen der Programm-Entwicklung, nachdem bisher die späteren Phasen, z.B. die Codierung (und damit die Programmiersprachen) ganz im Mittelpunkt gestanden hatten. Dabei wurde offensichtlich, daß eine Verbesserung der Qualität von Softwareprodukten sich nur dann erreichen läßt, wenn es gelingt, die Phasen, die vor der Programmierung liegen, zu systematisieren und zu rationalisieren, d.h. Hilfsmittel zur Verfügung zu stellen, die es ermöglichen

- die Anforderungen an die Problemlösung genau und in einer für die weitere Programmentwicklung geeigneten Form zu beschreiben,
- Mängel der Anforderungen zu erkennen,
- den Spezifikations- und Entwurfsvorgang ausreichend zu dokumentieren.

Ausgehend von diesen Überlegungen wurden Spezifikationssysteme entwickelt, die es versuchen, einen möglichst großen Teil des Programm-Lebenslaufs zu decken. Folgende Definition kann für ein Spezifikationssystem formuliert werden:
Ein Spezifikationssystem ist ein instrumentiertes und organisiertes Software-Entwicklungs-Laboratorium, in dem viele Personen arbeiten, um gemeinsam in einem vollständig organisierten Arbeitsprozeß Software zu entwerfen, zu konstruieren, zu prüfen, zu ändern und zu warten /1/.

### 2. Vergleich der Spezifikationssysteme

Die Spezifikationssysteme können nach folgenden Kriterien verglichen werden:

- Ideen
- Ansätze
- Art und Grad der Formalisierung

- Anwendungsbereich und dazugehörige Motivation und Ziele

## 2.1. Ideen

Die Spezifikationssysteme beruhen wesentlich auf folgenden Ideen:

| | |
|---|---|
| - spezielle Methoden der Softwareentwicklung | |
| Structured Design | AIDES |
| Jackson-Methode | DSL |
| - Sprachkonzepte | |
| CDL2 | CDL2-Laboratorium |
| Path-expressions | COSY |
| Pseudocode | TESO |
| - Management-Verfahren | |
| Projektlenkung | SDEM/SDSS |
| - besondere Ansätze | |
| besonderer Systembegriff (Computer-(Software-) System = Teilsystem in einem umfassenden soziotechnischen System) | DELTA/BETA |

## 2.2. Ansätze

Bei den Ansätzen für Spezifikationssysteme unterscheiden wir:

- homogener, heterogener Ansatz
- integraler, singulärer Ansatz
- aktionsorientierter, datenorientierter Ansatz

Ein homogener Ansatz ist ein solcher, der im wesentlichen um eine Technik gruppiert ist und der (in den meisten Fällen) auch historisch aus dieser Technik hervorgegangen ist.
Ein heterogener Ansatz entsteht durch Integration verschiedener gleichberechtigter Techniken, die nacheinander oder auch wahlweise für die verschiedenen Entwicklungsaufgaben eingesetzt werden können.
Bild 1 zeigt einen Vergleich zwischen homogenem und heterogenem Ansatz.

| | homogener Ansatz | heterogener Ansatz |
|---|---|---|
| Deckung des gesamten Software-life-cycle | schwer erfüllbar | gut erfüllbar |
| Flexibilität des Einsatzes | wird erschwert | wird erleichtert |
| Übergänge zwischen verschiedenen Entwicklungsphasen | leicht<br>einfach | schwer<br>kompliziert |
| Validation | wird erleichtert | wird erschwert |
| Beispiele | CDL2-Lab.,TESO | EPOS, HDM |

Bild 1

Bei dem integralen Ansatz wird jeweils eine Ebene als Ganzes betrachtet und beschrieben.
Bei dem singulären Ansatz betrachtet man nacheinander jeweils einen speziellen Aspekt und beschreibt alles, was mit diesem Aspekt zu tun hat.
Bild 2 zeigt einen Vergleich zwischen integralem und singulärem Ansatz./2/

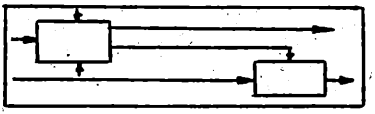
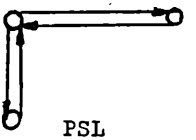
| integraler Ansatz | singulärer Ansatz |
|---|---|
| kompakte Beschreibung | aufwendige Beschreibung |
| übersichtliche Beschreibung | ohne automatische Dokumentationswerkzeuge unübersichtlich |
| Zusammenhänge sind sichtbar | Zusammenhänge nur durch automatische Analysen sichtbar |
| fördert ganzheitliche Problemsicht | fördert vereinzelte Problemsicht |
|  SADT |  PSL |

Bild 2

Bei dem aktionsorientierten Ansatz stehen die Aktionen im Mittelpunkt. Die von diesen Aktionen verarbeiteten Daten werden den Aktionen zugeordnet (z.B. HIPO).
Bei dem datenorientierten Ansatz stehen die Daten im Mittelpunkt. Zugriffsoperationen werden zugeordnet (z.B. Jackson-Methode).

## 2.3. Art und Grad der Formalisierung

Nach Art und Grad der Formalisierung lassen sich Spezifikationssysteme stark vereinfacht in natürlichsprachliche, programmiersprachliche und mathematische einteilen.
Im Bild 3 werden die drei Ansätze verglichen.

| | natürlich-sprachlich | programmier-sprachlich | mathematisch |
|---|---|---|---|
| Syntax formalisiert | nein | ja | ja |
| Semantik formalisiert | nein | nein | ja |
| Beispiele | PSI-Projekt-technik | EPOS, SADT, HIPO | RSL, Spezi, axiomatische Spezifikationen |

Bild 3

## 2.4. Anwendungsbereich und dazugehörige Motivation und Ziele

Man kann drei Anwendungsbereiche unterscheiden:

- Forschungsbereich
- industrieller Bereich
- öffentlicher Bereich

Motivation und Ziele des Forschungsbereiches sind es, Mittel zu schaffen, mit dem Methoden, Modelle, Verfahren, Techniken oder Werkzeuge für Software-Produktionsaufgaben laborartig entwickelt und erprobt werden können.
Im Forschungsbereich unterscheidet man wissenschaftliche (z.B. Entwicklung von Methoden zur Systemkonstruktion) und technische Ziele (z.B. Qualitätsverbesserung von Software).

Im Forschungsbereich werden homogene Spezifikationssysteme bevorzugt. Beispiele dafür sind COSY, DREAM, CDL2-Laboratorium. Motivation und Ziele des industriellen Bereichs sind die Handhabung großer Produktionsprojekte. Es werden heterogene Spezifikationssysteme bevorzugt, die stark durch die Branche beeinflußt werden, in der das jeweilige System eingesetzt wird. Es gibt im industriellen Bereich technische Ziele wie:

- Qualitätsverbesserung von Software,
- Berücksichtigung der Anwenderaspekte,
- Unterstützung von Methoden der Software-Entwicklung,

und es gibt auch wirtschaftliche Ziele wie:

- Kostenreduzierung,
- Produktionssteigerung,
- Produktionskontrolle.

Beispiele dafür sind CADES, AIDES.
Motivation und Ziele des öffentlichen Bereichs sind Standardisierung und Normung von Produkten und Produktionsverfahren, um bestimmte Qualitätsmerkmale zu sichern.

## 3. Werkzeuge

Wesentliche technische Komponenten der Werkzeuge sind:

- Sprachen für die Anforderungsdefinition und Systemspezifikation
- Instrumente und Verfahren zur Kontrolle und Verbesserung der Software-Qualität
- Werkzeuge zur graphischen Darstellung von Software
- Informationssysteme für technische Mitarbeiter und für das Management
- Werkzeuge für Produktionsplanung, Plankontrolle und Kontrolle des Produktionsprozesses

## 4. Einordnung von Verfahren

Bild 4 /1/ und 5 /1/ geben eine Übersichtseinordnung von Verfahren. Bei Bild 4 sind die Verfahren nach ihrer grundsätzlichen Methode eingeteilt, dagegen gibt Bild 5 eine Übersicht über existierende Software-Entwurfssysteme, die durchaus mehrere grundsätzliche Methoden unterstützen können (heterogene Systeme).

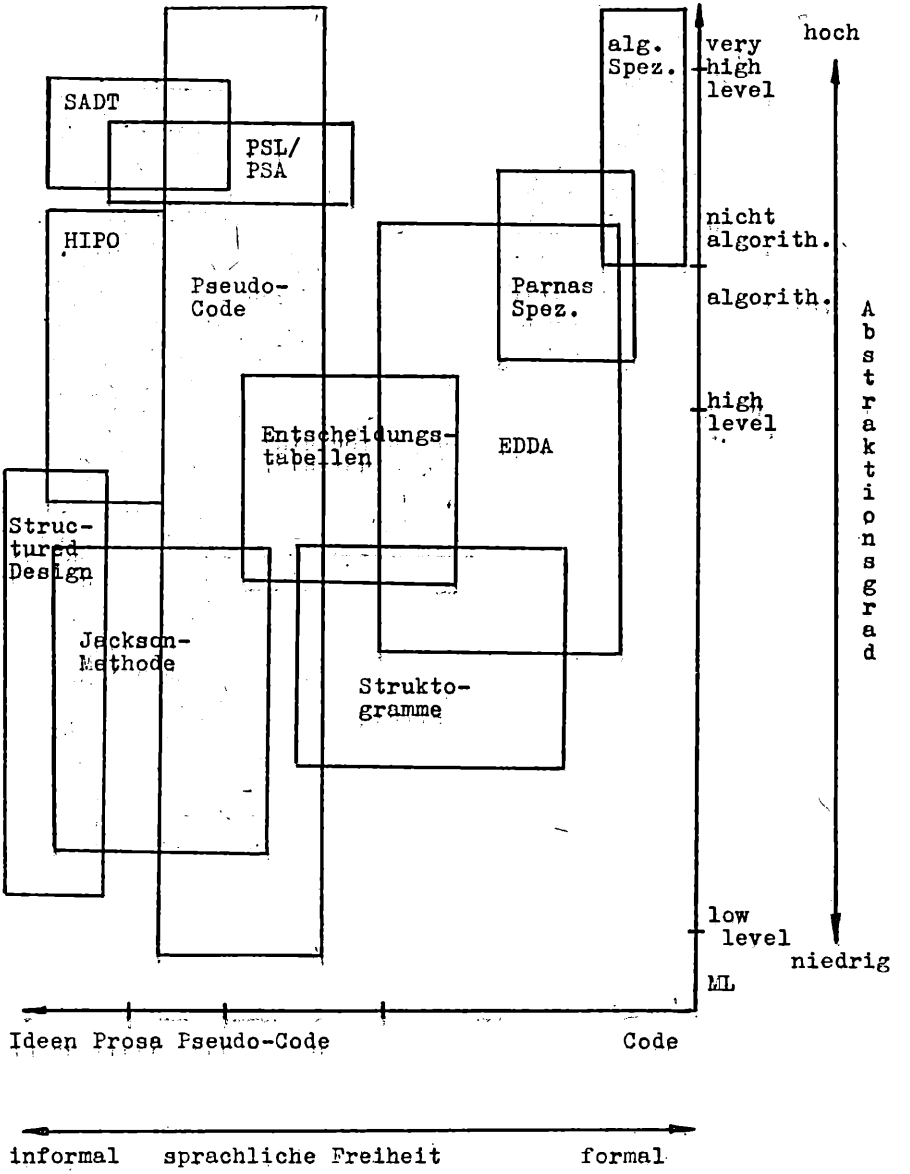

alg.
Spez.
very
high
level
hoch
SADT
PSL/
PSA
HIPO
Pseudo-
Code
nicht
algorith.
algorith.
Parnas
Spez.
high
level
Entscheidungs-
tabellen
EDDA
Struc-
tured
Design
Jackson-
Methode
Struktо-
gramme
Abstraktionsgrad
low
level
niedrig
ML
Ideen Prosa Pseudo-Code
Code
informal
sprachliche Freiheit
formal

Bild 4

| Spezifikationssystem-bezeichnung | Unterstützung der Produktion AN SE MI SI BW DF KE SS IN | Proj.-Fü. PL QS SZ BS | Bemerkungen, spez. Ziele, Anwendungen |
|---|---|---|---|
| CDL2 Compiler Description Language | CDL2 | | Übersetzerbau, System-Programm. |
| COSY Concurent Systems | COSY | | Betriebssystem-Entwicklung |
| CIP Computer-aided, Intuition-guided Progr. | CIP-L | | Programmentwickl. |
| PDS Program Development System | EL1 | | Programmentwickl. durch Transform. |
| HDM Hierarchical Development Methodology | HSL SPECIAL ILPL | | PARNAS-Spezifikationen |
| SDS Software Development System | RSL PDS | | große Militär-projekte |
| CADES Computer Aided Design Eval. System | SDL S3 | | Betriebssystem-entwicklung |
| AIDES Automated Interactive Design and Eval. System | SCG (SD) | DQMS | Flugzeugbau, große Graphiken |

———— Unterstützung

– – – – – – – – – teilweise bzw. geplante Unterstützung

Phasen:

AN: Analyse
DF: Definition
SE: Systementwurf
KE: Komponentenentwurf
MI: Modul-Implementierung
SS: Subsystem-Integration
SI: System-Integration
IN: Installation
BW: Betrieb & Wartung

Projektführung:

PL: Projektplanung
SZ: Aufwandsschätzung/-planung
QS: Qualitätssicherung
BS: Bewertung, Statistik

Bild 5

Auf dem Gebiet der Entwurfssysteme läuft gegenwärtig eine außerordentlich intensive Entwicklung ab, so daß eine Übersicht keinen Anspruch auf Vollständigkeit erheben kann. Ähnliches gilt auch für Fragen der Einordnung dieser Systeme.

Literaturhinweise

/1/ Goos,G. (Hrsg.), Werkzeuge der Programmiertechnik, Informatik-Fachberichte 43, Springer-Verlag, 1981

/2/ Floyd,Ch., Kopetz,H. (Hrsg.), Software engineering-Entwurf und Spezifikation, Teubner-Verlag Stuttgart, 1981

Verfasser:
Dr. Ing. Rolf Detering
TU Dresden
Sektion Informationsverarbeitung
8027 Dresden
Mommsenstr. 13

Dipl.-Ing. Valentin Stantschev
TU Dresden
Sektion Informationsverarbeitung
8027 Dresden
Mommsenstr. 13

Pfeiffer, Peter

# Modulkopplung durch interne Datenkommunikation

## 1. Datenstromnetzwerke

Der Begriff „Datenstromnetzwerk" (DSN) ist durch Weiterentwicklung der Grundzüge der datenorientierten Entwurfsverfahren entstanden.

Grundgedanke der datenorientierten Entwurfsverfahren ist die Widerspiegelung der Strukturen von Ein- und Ausgabedaten im logischen Skelett des Programms, das diese Daten verarbeitet bzw. erzeugt.

Zur Lösung von gewissen Strukturkonflikten (Begrenzungskonflikten), die zwischen den verschiedenen Datenstrukturen auftreten können, wird vorgeschlagen, durch vor- bzw. nachgelagerte Routinen (Unterprogramme, Coroutinen) Datenstrukturtransformationen vornehmen zu lassen. Weitere Komplikationen bei der automatischen Erzeugung des Programmskeletts entstehen durch die parallele Verarbeitung mehrerer Eingabedateien sowie durch die Realisierung von Backtrackingalgorithmen. Datenstromnetzwerke entstehen dadurch, daß man

- die Lösung von Begrenzungskonflikten durch Hintereinanderschalten von Moduln, die einen Strom von Eingangsdaten in einen Strom von Ausgangsdaten umformen, anstrebt (Arbeitsmoduln);
- Prozesse, die mehrere Eingabedatenströme zusammenfassen, in gesonderte Moduln verlegt, die für die nachfolgende Verarbeitung nur einen Ausgabestrom erzeugen (Mischmoduln);
- Prozesse, die aus einem Eingabedatenstrom verschiedene Ausgabedatenströme erzeugen, ebenfalls in speziellen Moduln konzentriert (Filtermoduln);
- Backtrackingprozesse durch spezielle Moduln realisiert, die aus einem Eingabedatenstrom einen "guten" Ausgabedatenstrom herausfiltern und an einem zweiten Ausgang einen "schlechten" Datenstrom bereitstellen;

- die Bereitstellung von Eingabedatenströmen speziellen Moduln überläßt, die die für die Kommunikation erforderlichen Datenstrukturen emittieren (Quellen);
- die Ausgabedatenströme in Moduln münden läßt, die die physische Ausgabe realisieren (Senken).

Eine aus den beschriebenen Moduln zusammengesetzte Modulstruktur wird hauptsächlich durch die durch die Moduln "fließenden" Datenströme beschrieben - die Modulfunktionen sind bei dieser Betrachtungsweise von nachgeordneter Bedeutung - aus diesem Grunde wurde für derartige Strukturen der Begriff Datenstromnetzwerk gewählt.

Zwischen den Moduln (Gliedern) eines DSN findet eine Übergabe von Datenstrukturen statt - und zwar ausschließlich in einer Richtung. Diese, die verschiedenen Gliedern eines DSN koppelnden Datenstrukturen, werden als interne Datenkommunikation bezeichnet. Die Daten, die ein DSN "durchströmen", "fließen" aus den Quellen über Misch-, Filter- und Arbeitsglieder in die Senken.

Sind DSN rückführungsfrei, d.h., die Ausgangsdatenstrukturen werden nicht an ein Glied als Eingangsstrukturen herangeführt, das die Voraussetzungen für das Erzeugen der Ausgangsstrukturen schafft, so handelt es sich um lebendige Netze - eine deadlock-Situation kann nicht eintreten. Derartige DSN sollen als von erster Ordnung bezeichnet werden.

DSN höherer Ordnung - sie enthalten Rückführungen und sind u.U. über Warteschlangen gekoppelt - müssen nicht notwendig lebendig sein. Derartige DSN sollten auf "deadlock-Freiheit" untersucht werden. Im folgenden werden derartige Strukturen und die mit ihnen zusammenhängenden Probleme nicht näher untersucht.

Datenstromnetzwerke werden vorteilhaft mit den Mitteln der Graphentheorie modelliert. Die Kanten widerspiegeln Datenströme (interne Datenkommunikation), die Knoten entsprechen den Gliedern des DSN. Ein DSN erster Ordnung wird demnach durch einen gerichteten, schlichten, kreisfreien, endlichen Graphen wiedergegeben (s. Bild 1).

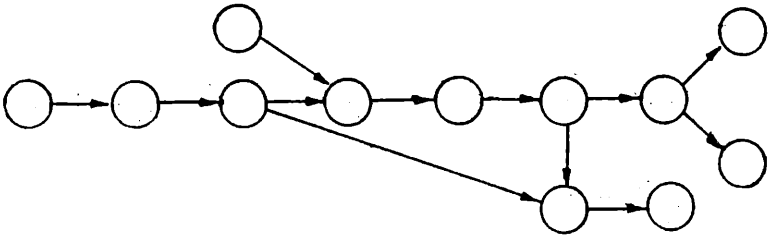


Bild 1: Wiedergabe eines DSN durch einen Graphen

Graphentheoretische Mittel wie

- Dekomposition
- Kondensation
  Vereinigung
- Durchschnittsbildung

und graphentheoretische Hilfsmittel wie

- Eingangsinzidenzmatrix
- Ausgangsinzidenzmatrix
- Knotenadjazenzmatrix

unterstützen den Entwurfs-, Implementierungs- und Testungsprozeß von DSN.

## 2. Klassifikation von Netzwerksknoten

Ordnet man den Knoten Ein- und Ausgangsvalenzen zu, so lassen sich folgende Knotentypen erkennen:

$V^+ = 0$, $V^- = 1$: Quellglieder
$V^+ = 1$, $V^- = 0$: Senkenglieder
$V^+ = 2$, $V^- = 1$: Glieder zur Datenstromzusammenführung
$V^+ = 1$, $V^- = 2$: Glieder zur Datenstromaufspaltung
$V^+ = 1$, $V^- = 1$: Arbeitsglieder.

In Bild 2 sind die aufgeführten Knotenklassen dargestellt - zusätzlich wurden einige Knotenklassen näher spezifiziert. Jedem der näher spezifizierten Knotenklassen (z.B. Mischglieder - MERGE) sind Klassen von Algorithmen zugeordnet, von denen ein Algorith-

mus ausgewählt werden muß, um einen realen Knoten implementieren zu können.

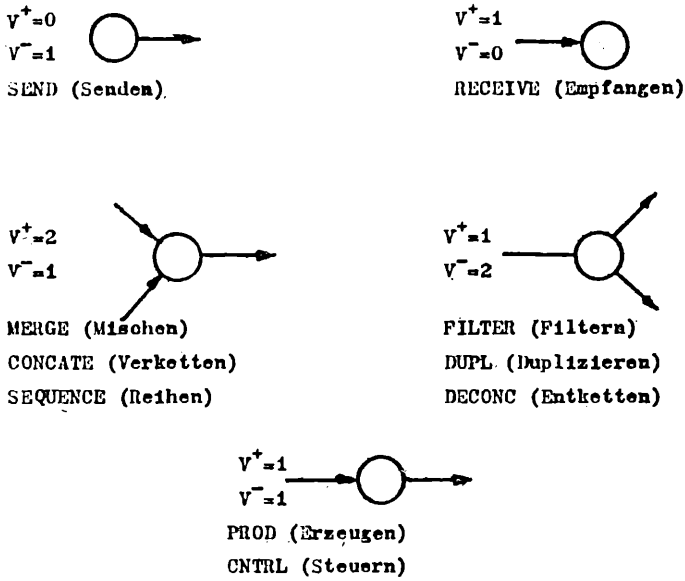



Bild 2: Übersicht über die Knotenklassen im graphentheoretischen Modell eines DSN

Neben den dargestellten Gliedern sind gegebenenfalls weitere zu entwerfen - beispielsweise spezielle Mischalgorithmen bei denen die Daten an den Eingabeschnittstellen über einen zeitgesteuerten Abtastalgorithmus sequentialisiert werden u.a.

Die Knotenklassen

Glieder zur Datenstromzusammenführung

und

Glieder zur Datenstromaufspaltung

(Standardglieder) werden mittels Makroinstruktionen realisiert, wobei die Algorithmenauswahl durch Makroparameter gesteuert wird.

Die Makroexpansionen führen zu hocheffektiven Assemblerprogrammen, die über die Schnittstelle "Datenstruktur" mit jedem in einer beliebigen Programmiersprache implementierten Nachbarmodul zusammenarbeiten können, sofern dieser die genannte Schnittstelle realisiert.

Die eigentliche Leistung eines DSN wird in den Arbeitsgliedern vom Typ PROD realisiert. Das logische Skelett derartiger Arbeitsglieder wird weitgehend automatisch aus den miteinander korrespondierenden Datenstrukturen generiert. Das Generierungsergebnis besteht aus einer Folge von Anweisungen in einer Zwischensprache und wird durch den Entwerfer präzisiert. Die Zwischensprache wird dann durch einen Prozessor in eine beliebige Zielsprache umgewandelt. Das Skelett schließlich wird durch die erforderlichen Aktionen (lineare Anweisungsfolgen) in der Zielsprache, in der das betreffende Arbeitsglied realisiert werden soll, komplettiert . Das das Arbeitsglied darstellende Programm wird anschließend mit einem Übersetzerprogramm für die Zielsprache übersetzt und gegebenenfalls mit der Gesamt- oder einer Teilstruktur zu einem Lademodul verbunden.

Für Quellglieder, Senkenglieder und Arbeitsglieder vom Typ CNTRL sind vorerst keine Entwurfs- und Implementierungshilfsmittel vorgesehen.

## 3. Kommunikative Datenstrukturen

Im graphentheoretischen Modell von DSN symbolisieren die Kanten Datenstrukturen, die der Modulkommunikation dienen (kommunikative Datenstrukturen).

Für die interne Datenkommunikation wird generell das variable Datenformat benutzt. Die Daten werden mittels geeigneter Koppelmechanismen (s. 4.) an den Modulschnittstellen übergeben. Da aus den Datenstrukturen in entscheidenden Netzwerksknoten die Logik des zugeordneten Moduls hergeleitet wird, muß es Möglichkeiten einer möglichst maschinenverarbeitbaren Beschreibung der Kanten und Knoten eines DSN geben.

Übersicht 1 gibt den ersten Entwurf einer deskriptiven Sprache für derartige Zwecke wieder.

```
programm::= datendefinitionsteil . relationenteil.
datendefinitionsteil::= {datendefinitionsanweisung}+
datendefinitionsanweisung::= 'DEF'.name.':'.datendefinition
datendefinition::= terminaldefinition | sequenz | alternative |
                   iteration | record | group | file
terminaldefinition::= (FIXED | FIXED SHORT | FLOAT SHORT |
                       FLOAT | FLOAT LONG | CHAR.{(.zahl .)}0:1 |
                       HEX.{(.zahl.)}0:1 | DECIMAL.{(.zahl.)}0:1 |
                       BOOLEAN|REFERENCE).'.'
sequenz::= '('.name.{','.name}+.').'
alternative::= '('.name.{';'.name}+.').'
iteration::= '('.name.'*'.zahl.{':'zahl}0:1.').'
record::= 'RECORD OF'.(sequenz|alternative|iteration)
group::= 'GROUP OF'.(sequenz | alternative | iteration)
file::= 'FILE OF'.(iteration|'('.name.',**).'|'('.name.',*+).')
name::= ...
zahl::= ...
relationenteil::= {relationsdefinition}+
relationsdefinition::= 'REL'.name.':'.relationsspezifikation
relationsspezifikation::= relationstyp.'('.{relationsoperand}1:3,.
relationstyp::= PROD|CNTRL|DUPL|FILTER.filtertyp|CONCATE |
                SEQUENCE|DECONC|MERGE.mergetyp|SEND|RECEIVE
relationsoperand::=
filtertyp::= ...
mergetyp::= ...
```

Übersicht 1: Erster skizzenhafter Entwurf einer deskriptiven Sprache zur Beschreibung von DSN

Die mittels einer derartigen Sprache erfolgende Spezifikation der Datenströme im DSN ist im allgemeinen ein iterativer Prozeß, der in eine dialogfähige Umgebung eingebettet werden muß. Die Generierung der Standardglieder und die Parserbereitstellung für die Arbeitsglieder erfolgt ebenfalls durch die Nutzung entsprechender Sprachkomponenten.

Das graphentheoretische Modell des DSN entsteht dabei sukzessive und kann das entscheidende Element der entwurfsbegleitenden Dokumentation bilden.

Die Schnittstelle "Datenstruktur" erlaubt es, die zu koppelnden Module in beliebigen Sprachen zu implementieren, wenn diese Sprachen Mittel zur Realisation der Koppelmechanismen enthalten oder nützen können.

## 4. Kopplungsmechanismen

Für die Kopplung von Moduln eines DSN werden folgende Koppelmechanismen eingesetzt:

- explizite Sequentialisierung (explicite sequencing) über
  - ein symmetrisches Coroutinenkonzept,
  - ein hierarchisches Coroutinenkonzept;
- implizite Sequentialisierung (implicite sequencing) über
  - ein MAILBOX-Konzept [CONS;MARL,KMQ,GRU].

Die implementierten oder in Implementierung befindlichen Konzepte sind den speziellen Erfordernissen der Ausbildung einer Schnittstelle zur internen Datenkommunikation angepaßt. An den Schnittstellen werden Datensätze der internen Kommunikation in Puffern bzw. Warteschlangen übergeben.

Die nachfolgenden Beispiele zur Illustration der Koppelkonzepte sind in einer Assemblernotation niedergeschrieben die einer realen Implementation (SKODA - [PFE]) entspricht. Bild 3 gibt die Kopplung über symmetrische Coroutinen wieder - solche Coroutinen sind zur Realisierung von Rückführungen vorgesehen. Die dort verwendete Makroinstruktion #CCALL realisiert den Aufruf einer symmetrischen Coroutine, die im PROGRAM-Makro als Coroutine aus-

gewiesen ist und dementsprechend generiert wird.

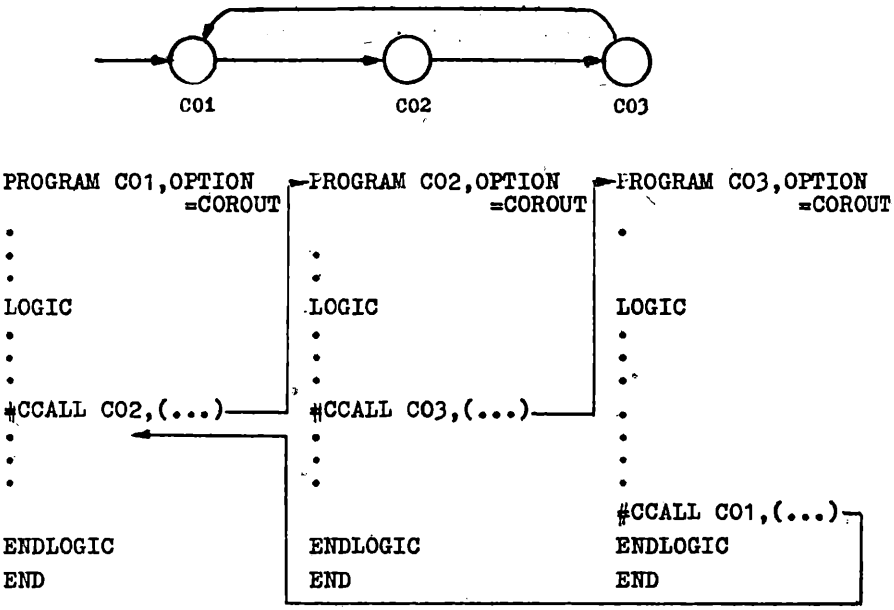
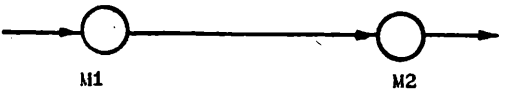



Bild 3: Modulkopplung über symmetrische Coroutinen

Bild 4 zeigt einen Anwendungsfall für das hierarchische Coroutinenkonzept. Hierin wird mit DCBC ein Steuerblock im Mastermodell generiert, über den die Modulkopplung erfolgt - durch #OPENC wird der Steuerblock eröffnet und durch #CLOSEC geschlossen. Die Datenkommunikation wird mittels der Makroinstruktionen #GETC bzw. #PUTC durchgeführt. Im nachgeordneten Modul, der über keinen Steuerblock verfügt, wird im befehlenden Makro der Parameter SLAVE kodiert.

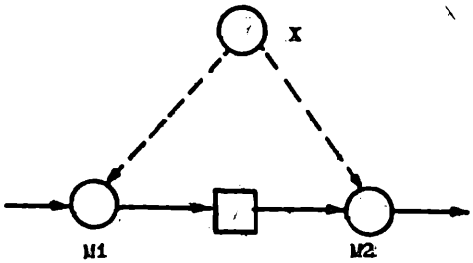



```
PROGRAM M1                          PROGRAM M2
•                                   •
•                                   •
•                                   •
DCBC PORT,PRODUCEL                  LOGIC
•                                   •
•                                   
•                                   
LOGIC                               
•                                   #GETC SLAVE,...
•                                   •
•                                   •
#OPENC (PORT,M2)                    •
•                                   ENDLOGIC
•                                   END
•
#PUT PORT,...
•
•

#PUTC PORT,...
•
•
•
#CLOSEC PORT
•
•
•
ENDLOGIC
END
```

Bild 4: Modulkopplung über das hierarchische Coroutinenkonzept

Bild 5 stellt die Modulkopplung über "implicite sequencing" dar. Den zu koppelnden Moduln wird ein Startmodul übergeordnet, der mittels der Makroinstruktionen STARTB, BOX und STARTE problemlos generiert werden kann. Aufgabe dieses Startmoduls ist der Start der dann asynchron arbeitenden zu koppelnden Moduln M1 und M2.




```
     STARTB X
     BOX PROD=M1,CONS=M2,EXPORT=PORT1,IMPORT=PORT2
     STARTE
```

```
PROGRAM M1                          PROGRAM M2
.                                   .
.                                   .
.                                   .
DCBM PORT1,PROD,LRECL=84,VOL=20     DCBM PORT2,CONS,LRECL=84,VOL=20
.                                   .
.                                   .
.                                   .
LOGIC                               LOGIC
.                                   .
.                                   .
.                                   .
#OPENM PORT1                        #OPENM PORT2
.                                   .
.                                   .
.                                   .
#PUTM PORT1,...                     #GETM PORT2,...
.                                   .
.                                   .
.                                   .
#CLOSEM PORT1                       #CLOSE PORT2
.                                   .
.                                   .
.                                   .
ENDLOGIC                            ENDLOGIC
END                                 END
```

Bild 5: Modulkopplung durch "implicite sequencing"

Analog zum hierarchischen Coroutinenkonzept (vgl. Bild 3) werden durch den DCBM ein Steuerblock in jedem der zu koppelnden Module generiert, der durch #OPENM eröffnet und durch #CLOSEM abgeschlossen wird.

Durch #GETM werden aus der durch die OPENM-Routine bereitgestellten Box Datensätze der internen Kommunikation entnommen, durch #PUTM werden derartige Sätze in die Box überstellt.

## 5. Implementierung - Stand und Absichten

Im Rahmen der Realisierung des Makrosystems SKODA [PFE] sind bisher implementiert

- die aufgeführten Coroutinenkonzepte,
- die Zwischensprache zur Strukturierung der Logik.

Im Stadium der Implementierung sind

- Makroanweisungen zur Generierung von Standardgliedern,
- Skelettgenerator.

Ferner wird an der Implementierung o.g. Komponenten für andere Zielsprachen (z.B. FORTRAN) gearbeitet.

Die Realisierung des gesamten Entwurfskonzepts ist bis 1985 geplant.

## Literaturhinweise

CONS: Constantine, L.L.: Control of Sequence and Parallelism in Modular Programs.
Proc. Spring Joint Computer Conference 1968
(A.F.I.P.S. Conference Proceedings, Bd. 32, S. 409-414)

GRU: Grune, D.: A View of coroutines.
A.C.M. SIGPLAN Notices, Bd. 12, Nr. 7 (July 77),
S. 75-81.

KMQ: Kahn, G. und MacQueen, D.: Coroutines and Network of Parallel Processes.
Information Proceedings 77, S. 993-998,
Gilchrift (Ed.), North Holland Publ. Co., Amsterdam 1977.

MARL: Marlin, C.D.: Coroutines. Lecture Notes in Computer Science 95,
Springer Verlin Berlin - Heidelberg - New York, 1980.

PFE: Pfeiffer, P.: Makrosystem SKODA - EDV-Aspekte,
Verlag Die Wirtschaft, Berlin, 3/82, S. 34-36.

Verfasser:

Dr. Peter Pfeiffer

Organisations- und Rechenzentrum
der Humboldt-Universität zu Berlin

1086 Berlin

Unter den Linden 6

PSF 1297

Richter, Eckhart
Tauscher, Wolfgang

# Zur Verwendung von Datenabstraktionen unter besonderer Beachtung der Belange der Assembler-Programmierung

## 1. Vorbemerkungen

In letzter Zeit wurde die Bedeutung der Abstraktion beim Entwurf von Software deutlich erkannt /3/. Unter Abstraktion versteht man eine allgemeine ingenieurtechnische Methode, um bei der Betrachtung komplexer Systeme und Wirkmechanismen die wesentlichen von den unwesentlichen Seiten zu trennen.
Erfahrungsgemäß besteht eines der größten Probleme beim Entwurf eines Softwaresystems darin, das komplexe Gesamtsystem in hinreichend einfache und verifizierbare Teilsysteme und Einzelprinzipien zu zerlegen. Alle Schritte bei einer solchen Dekomposition sind mit Abstraktionen verbunden, denen sich der Entwickler mehr oder weniger bewußt ist.

Nachfolgend werden einige Gesichtspunkte der Abstraktion angegeben, die bei der Aufgliederung eines Softwaresystems in der Entwurfsphase von Bedeutung sind. Außerdem werden eine Darstellungsmöglichkeit für die Systemarchitektur aufgezeigt und Werkzeuge für die Realisierung entsprechender Strukturen in Assemblersprache vorgeschlagen.

Die Autoren sind sich der Vorteile bewußt, die höhere Programmiersprachen gegenüber der Assemblersprache bieten. Tatsächlich verwendet und empfiehlt man die Assemblersprache /6/

bis auf den heutigen Tag für die Entwicklung und Weiterentwicklung von Betriebssystemkomponenten. Durch die Erweiterung der Assemblersprache um entsprechende Makros in Verbindung mit entsprechenden Programmierhilfen lassen sich alle wesentlichen, mit den modernen Programmiersprachen entstandenen Strukturen anwenden.

## 2. Abstraktionen in der Entwurfsphase

Der Entwurf eines Softwaresystems wird in 3 Schritte gegliedert:

(1) Systemspezifikation erarbeiten
(2) Systemarchitektur festlegen
(3) Algorithmierung.

Der erste Schritt für den Entwurf eines Softwaresystems ist eine möglichst vollständige Beschreibung des Verhaltens des beabsichtigten Systems in der definierten Umgebung. Diese Beschreibung wird auch als Systemspezifikation bezeichnet und könnte in Form einer Anwenderbeschreibung erstellt werden (Stufe E2 der PWT-Entwicklungsnomenklatur). Die Systemspezifikation ist im wesentlichen verbaler Art, sollte jedoch weitestgehend durch formale und graphische Darstellungen unterstützt werden (formale Syntaxbeschreibung, mathematische Funktionen, Ablaufpläne,...).
Mit der Systemspezifikation liegt fest, was das geplante System leisten soll. Die Systemspezifikation abstrahiert davon, wie die entsprechende Systemleistung realisiert werden kann (eigene Algorithmen und Daten sowie Nutzung der Umgebung).

Der nächste Entwurfsschritt besteht im Entwickeln der Systemarchitektur. Dabei wird das Gesamtsystem in Teilsysteme aufgeteilt. Diese Teilsysteme werden als Moduln bezeichnet. Zur Beschreibung eines Moduls gehören die im nachfolgenden Bild

angegebenen Informationen. Die Semantik der Funktionen und Daten wird in /1/ als Intended State Machines (ISM) bezeichnet.

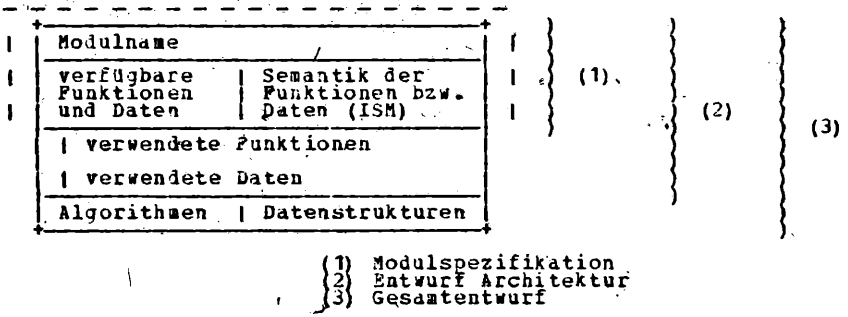



Bild 1. Betrachtungsweise eines Moduls

Einseitige informationelle Kopplung eines Moduls mit seiner Umgebung werden in Bild 1 durch parallele gestrichene und durchgehende Linien dargestellt.

Das Gesamtsystem ist vollständig aufgegliedert, wenn für alle Moduln die in Bild 1 genannten Informationen gewonnen sind.

Das Kriterium dafür, daß eine gewisse Menge von Einheiten (Funktionen, Daten) zu einem Modul zusammengefaßt werden, sind bestimmte, im betrachteten Zusammenhang wesentliche gemeinsame Eigenschaften der Einheiten. In diesem allgemeinen Sinne werden auch ADA-Packages gebildet /4/.

Das Problem dieser Aufgliederung besteht darin, die für die Architektur wesentlichen Gesichtspunkte zu erkennnen. Die Aufgliederung sollte adäquat den zu lösenden Problemen erfolgen. Erfahrungsgemäß wirkt sich eine ungünstige Systemarchitektur weit negativer auf die Qualität des Gesamtsystems aus, als Mängel bei

der Realisierung. Realisierungsmängel lassen sich auch einfacher beheben.

Für die Entwicklung der Systemarchitektur gibt es kein allgemeingültiges Rezept. Folgende Gesichtspunkte erscheinen wesentlich:

(1) Funktionelle Aufgliederung
Ein System kann entsprechend seinem Verhalten (ISM) in eine Menge von Teilfunktionen aufgeteilt werden. Die Teilfunkionen liefern jeweils einen Beitrag zur Realisierung der Gesamtfunktion. Diese funktionelle Aufgliederung wird soweit getrieben, bis Umfang und Kompliziertheit der Funktionen sinnvoll erscheinen.

(2) Aufgliederung nach Datenstrukturen
Wesentliche Daten, von nichttrivialer Struktur bilden zusammen mit den auf diese Daten definierten Funktionen eine abstrakte Datenstruktur, d.h. Abstraktion von der konkreten Struktur der Daten und von der Realisierung der entsprechenden Zugriffsfunktionen. Durch die Lokalität der Datenstrukturen ergeben sich die bekannten Zuverlässigkeitsvorteile.

(3) Abstraktion von Eigenschaften der Umgebung
Jedes Softwareprodukt arbeitet in einer bestimmten Umgebung. Die Informationen über die Umgebung sind in der Anwendungsprogrammierung i.a. in die Programmiersprache integriert. Die Programmiersprache enthält alle Funktionen der Umgebung, die ein Programm nutzen kann. Komponenten eines größeren Systems müssen die Funktionen der Umgebung durch geeigneten Aufruf nutzen.
Eine Komponente kann weitgehend von den Eigenschaften unabhängig gestaltet werden, indem zwischen Komponenten und Umgebung eine Schnittstelle definiert wird /5/. Kenntnis von der Umgebung erhalten ausschließlich gewisse

Schnittstellenmoduln. Durch diese werden die Anforderungen der entsprechenden Komponenten an die Umgebung in die funktionellen Möglichkeiten der konkret vorhandenen Umgebung transformiert.

Wenn die Komponente in einer anderen Umgebung genutzt werden soll, sind die Schnittstellenmoduln zu ändern. Dieser Gesichtspunkt der Abstraktion ist für Assemblerprogramme von besonderer Bedeutung.

(4) Ablaufsteuerung

Unter Ablaufsteuerung wird die Aufruffolge der einzelnen Moduln verstanden. Es sollte zunächst immer eine hierarchische Grundstruktur der Ablaufsteuerung angestrebt werden. Ein Modul ruft Funktionen hierarchisch nachgeordneter Moduln auf. Nach Beendigung der aufgerufenen Funktion geht die Steuerung an den rufenden Modul zurück.

Die hierarchische Grundstruktur wird durchbrochen, wenn mehrere Funktionen zu einem Modul zusammengefaßt werden. In diesen Fällen bleibt die Ablaufsteuerung linear, ist aber nicht mehr hierarchisch.

In weiteren Fällen:

- Fehlerbehandlung (Spezialfall: Abbruch nach Fehler)
- rekursive Probleme (z.B. Kommandointerpreter)

ist sogar eine nichtlineare Ablaufsteuerung der Problemlösung adäquat.

(5) Modulgröße, Modulkommunikation

Die einzelnen Moduln sollten weder zu klein noch zu groß sein. Zu kleine Moduln haben die Nachteile:

- Aufrufaufwand kann ins Gewicht fallen.
- Strukturbeziehungen zwischen den Moduln sind zu komplex.

Zu große Moduln haben die Nachteile:

- Modulinhalt ist zu komplex und unverständlich.

- Übersetzungszeiten sind in der Testphase zu hoch.

Das Problem besteht darin, einen sinnvollen Kompromiß zwischen der Komplexität der Systemarchitektur (Modulkommunikation) und der Komplexität der Modulinhalte zu finden.

Man muß ein möglichst objektives Maß für die Komplexität der Systemarchitektur und der Modulinhalte fordern. Man könnte sich dabei an den Ergebnissen in /7/ zur Programmkomplexität orientieren. U.a. wird dort das Maß 'Schwierigkeitsgrad' eines Programmes aus der Anzahl der Operationen und der Anzahl der Operanden abgeleitet.

Es wird folgende Darstellung von Modulstrukturen vorgeschlagen:

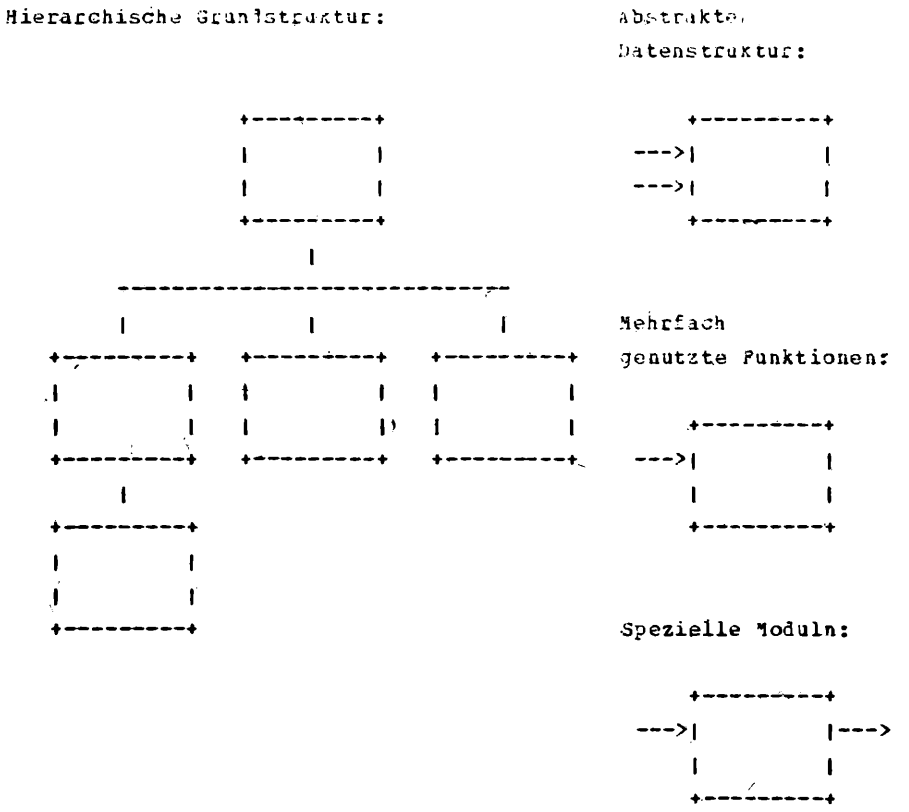



Bild 2. Darstellung der Systemarchitektur

Es sollte zunächst eine hierarchische Grundstruktur angestrebt werden. Diese bildet sich aus einer Menge von Funktionsmoduln, die

im Sinne von Unterprogrammen genutzt werden. Moduln, die an mehreren Stellen des Systems aufzurufen sind, sollten separat dargestellt werden.

Die Sprachen Modula-2 und ADA lassen dem Anwender beim Verknüpfen von Moduln (datenmäßig und aufrufmäßig) größte Freiheiten. Diese Möglichkeiten enthalten die Gefahr, daß ein Netz von Beziehungen zwischen den Moduln entsteht. Der Nutzer sollte auf eingeschränktere Möglichkeiten orientiert werden. Z.B. wäre es sinnvoll, Daten nur in Aufrufrichtung zu exportieren und die umgekehrte Richtung zu verbieten.

## 3. Realisierung in Assemblersprache

Für die speziellen Belange der Assemblerprogrammierung werden folgende Prinzipien bzw. Hilfsmittel vorgeschlagen:

- Die üblichen Makros zur Strukturierten Programmierung werden um folgende Makros zum Bilden und zum Aufruf von Moduln ergänzt:

```
fktname SUBR     Funktionsmodul bilden
-------------------------------------------------------------
clsname CLASS    Klassenmodul mit mehreren Funktionen bilden
  ...
fktname1 SUBR        Funktion 1
  ...
fktname2 SUBR        Funktion 2

-------------------------------------------------------------
CALL fktname      Funktionsmodul aufrufen
-------------------------------------------------------------
CLSCALL clsname,fktname
```

- Parameterübergabe bzw. Datenkommunikation zwischen verschiedenen Moduln erfolgt über einen gemeinsamen Datenmodul. Prinzipiell kann der Datenaustausch auch über Parameterlisten erfolgen. Wegen des höheren Aufwandes wird für die Kommunikation zwischen Moduln eines Teilsystems jedoch die Methode gemeinsamer Bereiche verwendet.

- Für den Fall, daß das entsprechende Softwaresystem reentrant sein soll teilen sich die Datenmoduln in einen gemeinsamen RO-Datenmodul und in einen RW-Datenmodul.

Literaturhinweise

/1/ Linger, R. C. The Management of software engeneering, Part III: Software design practices, IBM System Journal, Vol. 19/4/80

/2/ Morris, J. B., Programming by Successive Refinement of Data Abstractions, Software Practice and Experience, Vol. 10, No. 4, April 80

/3/ Horn, Ch., Die Spezifikationssprache ABSTRACT-PASCAL, RT/DV, Beiheft 3/81

/4/ Reference Manual ADA, 1980

/5/ Richter, E., Dialognutzer trainieren am Terminal, RT/DV, 1/82

/6/ Dyer, M., The Management of software engineering, Part IV: Software development practice, IBM System Journal, Vol. 19/4/80

/7/ Christensen, K., Filsos, G. P., Smith, C. P., A perspective on software science, IBM System Journal, Vol. 20/4/81

/8/ Wirth, N., Modula-2, Zürich, 80

Verfasser:

Dr.-Ing. Eckhart Richter
Dipl.-Ing. Wolfgang Tauscher
VEB Rechenzentrum Karl-Marx-Stadt
9010 Karl-Marx-Stadt
E.-Thälmann-Str. 7
PSF 312

Trautloft, R.

## Über einen Zugang zur automatisierten Auswahl physischer Realisierungen von relational definierten Bestandsdaten

1. Aufgabenstellung

Die hier vorzustellenden Ergebnisse entstanden im Rahmen von Arbeiten, die mit dem Ziel durchgeführt wurden, den Grad der Rechnerunterstützung im Softwareentwicklungsprozeß (SEP) zu erhöhen. Dabei erfolgte eine Beschränkung auf die für die Aufgabenklasse "programmtechnische Versorgung für automatisierte Informationssysteme" relevanten Aspekte. Für diese Aufgabenklasse nimmt die Teilaufgabe "Projektierung der Datenbasis" eine Schlüsselstellung im SEP ein. Das ist zum einen bedingt durch den Aufwand bei der Lösung dieser Aufgabe, zum anderen durch den Einfluß des Projektierungsergebnisses auf das Leistungsverhalten des entwickelten Softwaresystems in der Nutzungsphase.

In Abb. 1 ist ein Ausschnitt des SEP sowie die Einordnung markanter Entwicklungsstadien der Datenbasis dargestellt.

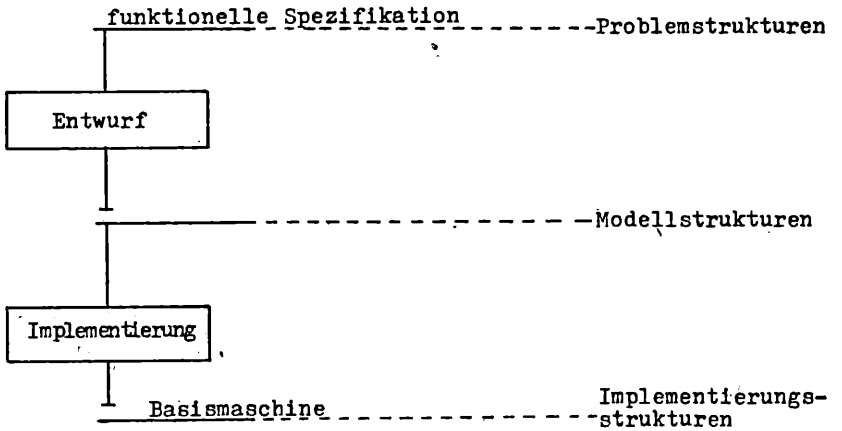



Abb. 1: Entwicklungsstadien der Datenbasis im Verlaufe des SEP

Die nachfolgenden Ausführungen beziehen sich auf den Aspekt der rechnerunterstützten Realisierung der Transformation von Modellstrukturen (M-Strukturen) in Implementierungsstrukturen (I-Strukturen).

2. M-Strukturen

Die in der Entwurfsphase des SEP zu realisierende Modellbildung wird durch die zur Verfügung stehenden sprachlichen Mittel beeinflußt. Die eingangs erwähnten Arbeiten haben, basierend auf der Programmiersprache PASCAL, zu einem Entwurfsbeschreibungsmittel geführt, das sich bezüglich der logischen Datendefinition am CODD'schen relationalen Datenmodell und bezüglich der Manipulation der so definierten Daten an der Sprache SEQUEL orientiert. Damit ist im Gegensatz zu den anderen bekannten Datenmodellen die Möglichkeit gegeben, streng zwischen logischer und physischer Datenbeschreibung zu trennen, d. h. in der Entwurfsphase implementierungsunabhängig zu arbeiten.

Für die Beschreibung der Datenbasis sind die M-Strukturen vom Typ TUPELSET und vom Typ RELATION von Interesse. Sie basieren beide auf dem TUPEL-Typ. Informal läßt sich dieser wie folgt charakterisieren:

- die Komponenten eines Tupels heißen Attribute
- Attribute sind vom einfachen Typ
- die Attribute eines Tupels setzen sich aus zwei disjunkten Teilmengen zusammen, der Menge der Schlüsselattribute und der Menge der Nichtschlüsselattribute.

Beispiel:

```
TYPE ANGEST = TUPEL [PERSNR:INTEGER] OF (NAME:ALPHA(15);
                    VNAM:ALPHA(8);PLZ:INTEGER;ORT:ALPHA(15));
VAR  PERS: TUPELSET OF ANGEST;
```

Es existiere weiter

```
VAR  FORSCH: TUPELSET OF THEMA;
```

zur Charakterisierung der Forschungsthemen einer Einrichtung, dann läßt sich die Mitarbeit der Angestellten an einem oder mehreren Forschungsthemen durch die Vereinbarung

```
VAR  ARBEIT: RELATION PERS WITH FORSCH OF THEMMIT;
```

beschreiben, wobei THEMMIT als Typ beispielsweise wie folgt vereinbart worden sei:

```
TYPE THEMMIT = TUPEL [PERSNR:INTEGER;THNR:ALPHA(5)] OF
                                          (VBE:REAL);
```

3. Automatisierungsansatz

Das Ziel besteht darin, die mit den eben vorgestellten sprachlichen Mitteln beschriebenen M-Strukturen in I-Strukturen der Basismaschine zu transformieren. Der Begriff der I-Struktur soll hier nur kurz mit dem Verweis auf Ähnlichkeit mit den häufig verwendeten Begriffen Speicherstruktur und Speicherzuordnungsstruktur eingeführt werden. Die Automatisierung dieses geistigen, im allgemeinen hochgradig kreativen Prozesses läßt sich als Lösung eines Auswahlproblems realisieren. Jede Implementierungsvariante D der Datenbasis besteht aus n I-Strukturen $K_{ij}$, d. h., die i-te M-Struktur wird durch die I-Struktur j implementiert. Dabei steht für die i-te M-Struktur ein I-Struktur-Vorrat von $m_i$ I-Strukturen zur Verfügung:

$$D = \{ K_{ij} \mid i = 1(1)n, \forall i: \exists j \in \{1,\ldots,m_i\} \}$$

Gesucht wird diejenige Implementierungsvariante, für die eine geeignete Bewertungsfunktion f Extremaleigenschaften besitzt:

$$\sum f(K_{ij}) = \text{extr}$$

Bei der Herleitung einer solchen Bewertungsfunktion bieten sich die Belastungsgrößen für die in der Nutzungsphase eingesetzten Ressourcen der EDVA als Auswahlkriterium an. Für die Softwaresysteme der betrachteten Aufgabenklasse sind der extern belegte Speicherplatz (S) und die Zahl der notwendigen externen Zugriffe (Z) zur Erfüllung der operationellen Anforderungen an die Datenbasis aussagekräftig für die Bewertung von I-Strukturen.

Die Verwendung solcher Bewertungskriterien impliziert die Notwendigkeit, Voraussetzungen für deren Quantifizierung zu schaffen. Zur Analyse der zu erwartenden operationellen Anforderungen an die I-Strukturen im Verlaufe der Nutzungsphase ist es möglich, eine Menge von elementaren Verarbeitungsoperationen anzugeben, auf die alle relevanten Anweisungen des Entwurfsbeschreibungsmittels zurückzuführen sind. Sie setzen sich aus Retrieval- und Update-Operationen zusammen. Weiter kann jede I-Struktur durch zwei analytische Modelle charakterisiert werden:

- Modell der Speicherplatzbelegung
- Modell des Zugriffsverhaltens in Abhängigkeit von der elementaren Verarbeitungsoperation.

Die durch konkrete Anforderungssituationen - beschrieben durch elementare Verarbeitungsoperationen - parametrisierbaren analytischen Modelle liefern somit eine quantitative Bewertung der I-Strukturen bezüglich der Kriterien S und Z. Die Formulierung eines Entscheidungsmodells zur Auswahl einer Implementierungsvariante der Datenbasis ergibt:

- Es existieren zwei Bewertungs- oder Gütekriterien (S, Z). Für beide ist ein Minimum anzustreben.
- Es existieren mehrere Steuergrößen:
  das externe Speichermedium
  die Realisierung des Primärzugriffspfades
  die Realisierung des Sekundärzugriffspfades.
- Die Abhängigkeit der Gütekriterien von den Steuergrößen ist durch die analytischen Modelle beschrieben.

Damit liegt eine Polyoptimierungsaufgabe vor /1/.

Zur Lösung der Polyoptimierungsaufgabe wurde ein heuristisches Lösungsverfahren gewählt. Die Heuristik wurde abgeleitet aus der Vorgehensweise des Softwareentwicklers im konventionellen SEP und basiert wesentlich auf der Ausnutzung der beiden Fakten:

- Antinomie zwischen S und Z
- Die Abarbeitungsfreundlichkeit eines Softwaresystems steigt mit dem Sinken der Zahl der benötigten externen Speichergeräte.

Zur Vorgabe eines Lösungsbereiches wird im Dialog der Softwareentwickler einbezogen.

4. Programmtechnische Realisierung

Abb. 2 zeigt die Struktur eines entwickelten Programmsystems, das den oben erläuterten Ansatz realisiert. Als Darstellungsmittel wurde ein Strukturdiagramm (nach CONSTANTINE "structure chart") verwendet. Es zeigt die als externe Prozeduren geschriebenen Bestandteile, ihre Steuerflußbeziehungen untereinander sowie die über Parameterübergabe realisierte Datenkopplung zwischen ihnen. Das Programmsystem umfaßt ca. 2500 Zeilen PASCAL-Text.

Sowohl bei der Formulierung des Automatisierungsansatzes als auch bei der programmtechnischen Realisierung wurde besonderer

Wert auf Erweiterbarkeit gelegt. Durch die Herleitung entsprechender analytischer Modelle ist der in die Auswahl einbezogene I-Struktur-Vorrat erweiterbar.

Für den Nachweis der Adäquatheit der zahlreichen Modellbildungen, auf denen der Automatisierungsansatz basiert, wurden einige I-Strukturen für eine über VSAM verfügende Basismaschine modelliert.

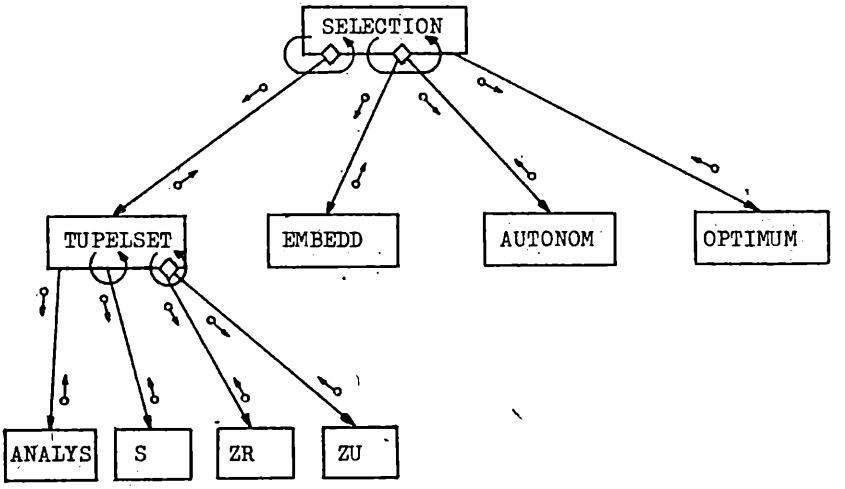



Abb. 2: Strukturdiagramm des Programmsystems

## 5. Schlußbemerkungen

Die programmtechnische Realisierung weist die Anwendbarkeit des Automatisierungsansatzes nach. Damit wurde eine wesentliche Vorleistung für eine in ein System zur rechnerunterstützten Entwicklung von Softwaresystemen (der gegebenen Aufgabenklasse) /2/ integrationsfähige Komponente erbracht.

Außerhalb eines solchen Systems könnten die erzielten Ergebnisse die Basis bilden für die Entwicklung eines autonom nutzbaren Werkzeuges für den SEP. Dieses Werkzeug stände als Entscheidungshilfe für den Softwareentwickler zur Verfügung und würde die für andere technische Entwicklungsprozesse typische und bewährte Arbeitsweise des Ingenieurs mit dem Variantenvergleich ermöglichen.

Literaturhinweise

/1/ Trautloft, R.: "Selektion von Implementierungsstrukturen - ein Automatisierungsansatz für die Implementierung der Datenbasis von Softwaresystemen" 3. Symposium "Grundlagen und Anwendung der Informationsverarbeitung", Karl-Marx-Stadt, 1982, in: "edv-aspekte" 1 (1982) 3

/2/ Lindner, U.: Zugang zum rechnerunterstützten Entwurf und der Automatisierung von Implementierungsentscheidungen
bit' 82, Berlin, 1982

Verfasser:
Dipl.-Ing. Rainer Trautloft
TH Karl-Marx-Stadt
Sektion Rechentechnik/Datenverarbeitung
DDR 9001 Karl-Marx-Stadt
PSF 964

Polze, Christoph

# Experimente mit der Programmiersprache CDL2

## 1. Einleitung

Die Programmiersprache CDL2 wurde als Hilfsmittel zur Implementation von Compilern entwickelt /1,2/. Sie hat sich dabei bestens bewährt. Naturgemäß ist jedoch auch ihre Verwendung als Ausgangspunkt für das syntax-orientierte Programmieren überhaupt, wie es durch die Jacksonsche Entwurfsmethode empfohlen wird /3/. Im folgenden Vortrag wird aus einem der Standardbeispiele aus der Welt der Jacksonschen Entwurfsmethode ein CDL2-Programm hergeleitet. Es zeigt sich, daß bereits in einem sehr frühen Entwurfsstadium verarbeitungsfähige CDL2-Programme entstehen. CDL2 erweist sich dabei gleichsam als ein Werkzeug zum programmiersprachlich geführten top-down-Entwurf.

## 2. Aufgabe /4/

"Eine Lochkartendatei ist aufsteigend sortiert nach einem Schlüssel, der in jeder Karte enthalten ist. Innerhalb dieser Reihenfolge ist die erste Karte jeder Gruppe mit einem gemeinsamen Schlüsselwort eine Kopfkarte, die restlichen sind Detailkarten. Jede Detailkarte enthält ein numerisches Betragsfeld. Es soll ein Bericht erstellt werden, der die Gruppensummen für alle Schlüsselworte ausweist".

## 3. Jacksonscher Entwurf

Es wird eine formale Beschreibung der Eingabedatenmenge und eine formale Beschreibung der Ausgabedatenmenge benötigt. Der Programmentwurf besteht darin, Beziehungen zwischen Eingabebeschreibung und Ausgabebeschreibung herzustellen.

Beschreibung der Eingabedaten:

```
card file: group, *.
group: key card, group tail.
key card: . #Test auf Schlüsselkarte#
```

```
group tail: detail card, *.
detail card: . #Test auf Detailkarte#
```

Beschreibung der Ausgabedaten:

```
report: title, report tail.
title: , #irgendeine Überschrift#
report tail: line, *.
line: initialize, result.
initialize: . #Bestimmung eines Anfangswerts#
result: accumulate, *.
accumulate: . #irgendeine Verknüpfung#
```

Zur Verbindung von Eingabedaten und Ausgabedaten werden folgende Verbindungsoperationen benötigt

(a, ident, b): Identifiziere die Variable a aus der Eingabedatenbeschreibung mit der Variablen b der Ausgabedatenbeschreibung

(, incl, b): Die Variable b der Ausgabedatenbeschreibung wird ohne Bezug auf die Eingabedaten in das Entwurfsprodukt einbezogen.

(a, seq, b): Das entworfene Programm enthält die Variable a der Eingabedatenbeschreibung und die Variable b der Ausgabedatenbeschreibung einzeln in der Reihenfolge a vor b.

Ausgehend von den Datenbeschreibungen des Beispiels werden als Variablenverbindungen vorgesehen

```
(card file, ident, report)
   (, incl, title)
   (, incl, report tail)
(group, ident, line)
(key card, seq, initialize)
(group tail, ident, result)
(detail card, seq, accumulate)
```

Daraus entstehen unmittelbar eine eingabeorientierte und eine

ausgabeorientierte Fassung des entworfenen Programms.

Eingabeorientierte Fassung:

```
card file: title, report tail.
title: .
report tail: group, * .
group: key card, initialize, group tail.
key card:
initialize: .
group tail: detail card, accumulate, * .
detail card:
accumulate:
```

Ausgabeorientierte Fassung:

```
report: title, report tail.
title: .
report tail: line, * .
line: key card, initialize, result.
key card:
initialize: .
result: detail card, accumulate, * .
detail card:
accumulate:
```

Beides sind bis auf etwas "CDL-Zucker" fertige verarbeitungsbereite CDL-Programme.

Es müßten sich Entwurfsschritte zur Dateibehandlung, eof-Test, Fehlerbehandlung und zur Festlegung der Verknüpfungsoperationen anschließen.

## 4. CDL2 Fassung des Programms

Für die eingabeorientierte Fassung des Programms wird ein vollständiges Programm angegeben. Es enthält im Abschnitt mit der Bezeichnung basic einen Testrahmen. Das obige Entwurfsprodukt findet man im Abschnitt S.

```
LAYER 1.
SECTION basic.
EXT is, zero, one.
CONST zero=63, one=64.
VAR first character.
PREDICATE is +>x: equal+x+first character, next card.
ACTION nextcard: newline+input,
                 outchar+output+first character,
                 newline+output,
                 inchar+input+first character.
CONST input=0, output=10.
ACTION inchar+>channel+x>=
       "$L$T EXTNCALL $T INCHAR".
ACTION outchar+>channel+>x=
       "$L$T EXTNCALL $T OUTCHAR".
ACTION newline+>channel=
       "$L$T SLR $T "X", "Y.
PRELUDE inchar+input+first character.
ENDSEC basic.

SECTION s.
INV is, zero, one.
ACTION card file: title, report tail.
ACTION title: .
ACTION report tail: group, ж; .
PREDICATE group: key card, initialize, group tail.
PREDICATE key card: is+key.
CONST key =one.
ACTION initialize: .
ACTION group tail: detail card, accumulate, ж;.
PREDICATE detail card: is+detail.
CONST detail=zero.
ACTION accumulate:.
ROOT card file.
ENDSEC s.

ENDLAY 1.
PRELUDE basic.
ROOT s.
```

Auszüge aus dem Druckprotokoll des CDL2-Compilers von KU Nijmegen Version 8.1. (79-05-03) belegen das Testergebnis:

```
DIAGNOSTICS FROM PHASE 1
***** 0  SEVERE ERROR(S)
  +++ 0  ERROR(S)
  --- 0  WARNING(S)

DIAGNOSTICS FROM PHASE 2
***** 0  SEVERE ERROR(S)
  +++ 0  ERROR(S)
  --- 0  WARNING(S)

    1
    0
    0
    1
    0
    0
    0
```

Literaturhinweise

/1/ C.H.A. Koster: Using the CDL compiler-compiler in: Compiler construction, an advanced course, Lecture Notes in Computer Science 21, Springer Verlag, Berlin 1976

/2/ J.P. Dehottay, H. Feuerhahn, C.H.A. Koster, H.M. Stahl: Syntaktische Beschreibung von CDL 2, Forschungsprojekt CDL2, Informatik, Technische Universität Berlin, 1976

/3/ M.A. Jackson: Principles of program design, Academic Press,1975

/4/ Robotron ZFT: Systemunterlagen-Dokumentation OS/ES. Handbuch der POS-Technologie 10/80

Verfasser:
Doz. Dr. sc. nat. Christoph Polze
Humboldt-Universität zu Berlin
Sektion Mathematik
1086 Berlin, Postfach 1297

Bothe, Klaus

## MODULARE PROGRAMMIERUNG IN PASCAL (Kurzfassung)

Die Programmiersprache PASCAL /1/ besitzt außer dem Prozedurkonzept keinerlei Möglichkeiten zur modularen Strukturierung von Programmen, wie sie in neueren Programmiersprachen bestehen.
So verfügen z.B. Ada /2/ und Modula-2 /3/ über die Ausdrucksmittel des package und des module, die äußerlich eine Sammlung von Definitionen und Deklarationen darstellen. Mit Hilfe dieser Mittel kann u.a. das Prinzip des abstrakten Datentyps /4/ realisiert werden, indem eine Menge von verwandten Operationen und die gemeinsame Datenmenge, über der sie operieren, zusammengefaßt werden.
Die Zielstellung bestand nun darin, diese modularen Konzepte durch eine geeignete Software-Unterstützung in die PASCAL-Programmierung einzubeziehen. Hierzu wird eine Spracherweiterung von PASCAL, M-PASCAL, vorgeschlagen: PASCAL wird um das Module-Konzept von Modula-2 erweitert.
Zur Unterstützung von M-PASCAL wurde ein Precompiler, der M-PASCAL in PASCAL überführt und ein Compiler für M-PASCAL implementiert.

Literatur:

/1/ N.Wirth: Revidierter Bericht über die Programmiersprache PASCAL, Akademieverlag 1976.

/2/ The programming language Ada, LNCS 106, 1980.

/3/ Wirth: MODULA-2, ETH Zürich, Bericht 36, 1980.

/4/ K.Bothe:Restructuring a compiler by abstract data types, Seminarbericht Nr.40, Sektion Mathematik, Humboldt-Univ.

Dr. Klaus Bothe
Sektion Mathematik
Humboldt-Universität zu Berlin
1086 Berlin, PSF 1297

Hohberg, Bodo

## Syntaxorientierter Programmeditor für PASCAL

### 1. Einleitung

An der Sektion Mathematik, Bereich IV, laufen zur Zeit Arbeiten an einem Softwarelabor für die Programmiersprache PASCAL. Als erste Teilaufgabe wird ein syntaxorientierter Programmeditor für PASCAL realisiert. In Ergänzung zu den üblichen Funktionen eines Texteditors können durch einen syntaxorientierten Programmeditor folgende Aufgaben realisiert werden:

- Syntaxkontrolle
- Adressierung fehlerhafter Syntaxeinheiten(Zeilen)
- Anfertigen von Strukturübersichten
- formatierte Quelltextausgabe
- komplexe Operationen mit Syntaxeinheiten

Es erfolgt keine Syntaxführung des Nutzers wie z.B. in /1,2/. Syntaxkontrollen erfolgen nur auf Kommando des Nutzers. Die Realisierung greift auf einige Ideen aus /3/ zurück (Aufbau eines Syntaxbaumes) und kann in einigen weiteren Punkten mit /4/ verglichen werden. Die Implementation erfolgte in PASCAL.

Die Nutzung im TSO des OS/ES für Quelltextbücher bis 800 Zeilen Länge ist bis Ende 1982 vorgesehen. Eine Überführung auf K1620 und K1520 ist für das erste Halbjahr 1983 geplant.

### 2. Arbeitsweise des Editors

Syntaxorientierte Editoren sind für die Dialogarbeit bestimmt. Für die Auswahl von Syntaxeinheiten benötigt der Nutzer laufend Informationen über die aktuelle Struktur des editierten Quelltextes. So sind folgende Dialogschritte vorgesehen:

- Ausschrift über die Struktur des Programms
- Adressierung einer zu bearbeitenden Einheit
- Änderungen an der adressierten Einheit

Nach jedem Kommando erfolgt eine Ausschrift, die sowohl als Adressierungshilfe als auch zur Kontrolle der richtigen Funktion des vorangehenden Kommandos gedacht ist.
Die zur Verfügung gestellten Kommandos untergliedern sich in fünf Gruppen:

1. Adressieren einer neuen Einheit
2. Eröffnen bzw. Abschließen von Syntaxeinheiten zum Editieren
3. Ändern des Inhalts der editierten Syntaxeinheit
4. Textoperationen
5. Analyseoperationen

Die Möglichkeit, das Editieren auf einzelne Syntaxeinheiten einzuschränken, soll in Verbindung mit der strukturierten Programmierung die Bildschirmarbeit übersichtlicher machen helfen.

## 3. Einheiten, Folgenummern

Einheiten, die sich durch den Editor adressieren und bearbeiten lassen, sind ausgezeichnete Syntaxeinheiten und Zeilen. Bei den ausgezeichneten Syntaxeinheiten wird unterschieden zwischen:

1. zusammengesetzten Syntaxeinheiten:
   Programm, Prozedur, Funktion, Modul (Erweiterung entsprechend /5/)
2. elementaren Syntaxeinheiten:
   Köpfe der zusammengesetzten Syntaxeinheiten (einschließlich der Markendeklarationen), Konstanten-, Typdefinitionsteile, Variablen-, Routinendeklarationsteile und Verbundanweisungen

Die elementaren Syntaxeinheiten sind für den Editor Zeilenfolgen. Die zusammengesetzten Syntaxeinheiten sind Folgen der unmittelbar enthaltenen Syntaxeinheiten.
Zeilenfolgen und Syntaxeinheitenfolgen werden mit 1 beginnend durchnumeriert. Die zugeordneten Nummern heißen Folgenummern. Die Folgenummern dienen als Adressierungshilfe. Nach jeder Änderung werden sie neu zugeordnet und durch die Ausschrift des Editors sichtbar gemacht. Zur Bezeichnung der zusammengesetzten Syntaxeinheiten wird der Name der Syntaxeinheiten benutzt, sonst das erste Morphem.
Es sei das folgende Programm gegeben:

```
PROGRAM SUMME(INPUT,OUTPUT);
VAR I,J,SUMM:INTEGER;
```

```
BEGIN
   READLN(INPUT,I);
   FOR J:=1 TO I DO SUMM:=SUMM+J;
   WRITELN(' SUMME DER ZAHLEN VON 1 BIS',I:3,' =',SUMM:6);
END.
```

Das Programm SUMME ist eine zusammengesetzte Syntaxeinheit. Es besteht aus der Folge der elementaren Syntaxeinheiten

1_PROGRAM, 2_VAR, 3_BEGIN

Der durch 3_BEGIN bezeichnete Verbund besteht aus der Folge der Zeilen

```
1_BEGIN
2_   READLN(INPUT,I);
3_   FOR ...
4_   WRITELN...
5_END.
```

(Das an die Folgenummern angefügte '_' Zeichen macht diese als Folgenummern kenntlich.)

## 4. Adressierung von Einheiten

Die Adressierung von Einheiten erfolgt nach dem Kursorprinzip. Der Nutzer gibt die Nummer einer Einheit an. Der Editor ordnet der adressierten Einheit in der Ausschrift '+++' zu und macht sie so kenntlich. Durch die Eingabe von Null kann der Kursor vor die erste Einheit gesetzt werden.

Beispiele:

Adressierung der 2.Einheit im Programm SUMME

```
:2
1_PROGRAM, +++VAR, 3_BEGIN
```

(Die Eingabe des Nutzers wurde unterstrichen. ':' zeigt dem Nutzer eine Kommandoanforderung.)

Adressierung der 3.Einheit im Verbund, wenn der Verbund die editierte Einheit ist

```
:3
2_   READLN(INPUT,I);
+++  FOR...
4_   WRITELN...
```

Die Ausgabe des Kontextes der adressierten Einheit erleichtert die Kontrolle der Adressierungsoperationen.
Weitere Kommandos zur Adressierung sind:

(+/-) [n] Vor- bzw. Zurücksetzen des Kursors um n bzw. 3 Einheiten

Beispiele:

+4 Weitersetzen um 4 Einheiten
- Zurücksetzen um 3 Einheiten

## 5. Eröffnen von Einheiten zum Editieren und Abschließen der Einheiten

Nach dem Aufruf des Editors kann durch das Kommando INPUT ein Quelltextbuch eingelesen werden, das vorher dem PRD-File zugeordnet wurde. Es erfolgt nach dieser Eingabe eine Strukturanalyse und Ausgabe einer Übersicht über die Syntaxeinheiten auf dem äußersten Niveau. Erfolgt z.B. die Eingabe des Programms SUMME vom PRD-File, so gibt der Editor als erstes folgendes aus:

```
! SUMME.1_
+++ PROGRAM, 2_VAR, 3_BEGIN
```

Es ist die umfassendste Syntaxeinheit zum Editieren eröffnet. Die erste Einheit ist adressiert. Die enthaltenen Einheiten können im Direktzugriff als unzerlegbare Einheiten über die Folgenummern wie Zeilen in Texteditoren ersetzt, gelöscht, an eine andere Stelle transportiert werden usw. Soll jedoch der Inhalt der Einheiten schrittweise verändert werden, so sind Syntaxeinheiten zum Editieren zu eröffnen. Nach dem Eröffnen wird eine Information über die Folge der darin enthaltenen Einheiten ausgegeben, die nun durch den Editor anzusprechen sind.
Z.B. läßt sich nach der obigen Ausgabe der Verbund zum Editieren eröffnen:

```
:3.
! SUMM.BEGIN.1_
+++ BEGIN
2_      READLN(INPUT,I);
3_      FOR J:=1 TO I DO SUMM:=SUMM+J;
```

Durch das Kommando 3. wurde die 3.Einheit adressiert und zum Editieren eröffnet. Ein zeilenweises Editieren des Verbundes

ist nun möglich.
Zum Abschließen von Syntaxeinheiten existiert das Kommando '*'.
Bei Forsetzung des Beispiels mit dem Kommando '*' ergibt sich:

```
:*
! SUMM.1_
+++ PROGRAM, 2_VAR, 3_BEGIN
```

Die editierte Einheit wird abgeschlossen. Das Editieren der umfassenden Einheit läßt sich fortsetzen.
Der Editor gibt zur besseren Orientierung bei diesen Operationen zusätzlich die Namen der umfassenden Syntaxeinheiten aus.
Zum Abschließen mehrerer umfassender Syntaxeinheiten wurde das Kommando

* Name

implementiert. Es werden alle umfassenden Syntaxeinheiten einschließlich der Syntaxeinheit mit dem angegebenen Namen abgeschlossen. Wird die umfassendste Syntaxeinheit (z.B. das Programm SUMME) abgeschlossen, so ist das Editieren beendet. Es erfolgt eine Ausgabe des Quelltextes auf das zuzuordnende PRR-File.
(Für die Dialogeingabe eines Quelltextes ist das Kommando :=TE zu verwenden, vgl. Abschn.6.)

## 6. Ändern des Inhalts der editierten Syntaxeinheit

Jede editierte Syntaxeinheit besteht aus einer Folge von Syntaxeinheiten oder Zeilen. Objekt einer Operation kann eine Einheit (gegeben durch eine Folgenummer) oder eine Teilfolge von Einheiten sein. An Operationen sind zulässig:

| | |
|---|---|
| OBJ1:=OBJ2 | Wertzuweisung |
| OBJ1:=' | Löschen |
| OBJ1<=OBJ2 | Transport von OBJ2 auf die Stelle von OBJ1 |
| OBJ1<=>OBJ2 | Vertauschen |
| OBJ1#OBJ2 | Verkettung |

Kodierung der Operanden:

| | |
|---|---|
| leer | adressierte Einheit |
| n1..n2 | Folge der Einheiten n1,...,n2 |
| n | n-te Einheit |

Ein einzugebender Operand wird durch TE symbolisiert.
Nach jeder Operation wird ein überschriebenes Objekt im Bereich SAVE gerettet. Durch SAVE läßt sich dieses Objekt im folgenden adressieren. TE und SAVE sind nur als OBJ2 zulässig.
Bei der Verkettung wird die Stelle vor der ersten Einheit durch Null adressiert.
Beispiele für Kommandos:

| | |
|---|---|
| 2:=' | 2.Einheit löschen |
| 1#SAVE | gerettete Einheit wieder einfügen |
| 1<=4 | 4.Einheit an die Stelle der ersten Einheit transportieren |
| 2<=>3..4 | 2.Einheit hinter 4.Einheit transportieren |
| 2:=TE | 2.Einheit durch einzugebenden Text ersetzen |

Die Texteingabe beginnt nach der Ausschrift

```
! WRITE TEXT UNTIL / *
```

und endet nach der Eingabe von / * am Anfang einer neuen Zeile.

## 7. Textspeicher

Um komplexe Umstrukturierungsoperationen an Programmen ausführen zu können (wie sie z.B. bei größeren Änderungen erforderlich sind), wurden drei Textspeicher zur Verfügung gestellt. Sie tragen die Bezeichnungen

TEXT1, TEXT2, TEXT3.

Alle Editieroperationen, die für Programme vorgesehen sind, können nach dem Eröffnen eines Textspeichers zum Editieren (Kommando TEXT1 oder TEXT2 bzw. TEXT3) auch auf den im Textspeicher enthaltenen Text bzw. enthaltene Folge von Syntaxeinheiten angewendet werden.
Alle Textspeicher können beliebig als Operand bei den Änderungsoperationen auftreten.
Beispiele:
In TEXT1 ist eine neue Programmversion eines bereits geladenen Programms zusammenzustellen. Die Struktur des Programms wird durch die Ausschrift des Rechners sichtbar:

```
! BEISPIEL1.1_
+++ PROGRAM, 2_CONST, 3_TYPE, 4_HILFSPROZ, 5_BEGIN
```

Folgende Aktionen sind jetzt denkbar:

Eingabe des neuen Programmkopfes:

```
:TEXT1:=TE
! WRITE TEXT UNTIL / *
1 PROGRAM BEISPIEL2;
2 / *
```

Anfügen der 2. Syntaxeinheit des gegebenen Programms:

```
:TEXT1#2
```

Anfügen der 4. und 5.Syntaxeinheit des gegebenen Programms:

```
:TEXT1#4..5
```

Eröffnen von TEXT1 zum Editieren, um noch auszuführende Änderungen durchzuführen:

```
:TEXT1
```

Ausführung der Änderungen ...
Abschließen von TEXT1 (Das alte Programm wurde bisher nicht verändert):

```
:*
! BEISPIEL1.1_
+++ PROGRAM, 2_CONST, 3_TYPE, 4_HILFSPROZ, 5_BEGIN
```

Überschreiben des alten durch das neue Programm:

```
:1..5:=TEXT1
```

Abschließen des Editierens:

```
:*
```

Im Rahmen des noch zu implementierenden Softwarelabors ist an eine Globalisierung der Textspeicher gedacht, so daß nach Abschluß des Editors in TEXT1 weiter die neue Programmversion zur Verfügung steht und über die Weiterverwendung (Speicherung) später entschieden werden kann.

## 8. Textoperationen

Jede vom Editor verwaltete Einheit besitzt als Wert eine Quelltextzeile oder eine Folge von Quelltextzeilen. Es sind Operationen vorgesehen, mit denen man im Text einer Einheit einen Bezeichner oder eine Zeichenkette

- suchen
- beim ersten Auftreten ersetzen
- bei jedem Auftreten ersetzen kann.

Als Kommandoform ist vorgesehen:

SEARCH Op

(ALTER/ALTERALL) Op Op

Ein Operand Op hat die Form:

␣Bezeichner oder

z Text z mit z Zeichen ≠ leer und Text Folge von Zeichen, die z nicht enthält.

Für zwei aufeinanderfolgende Texte gilt: z Text1 z Text2 z

Ein Bezeichner wird entsprechend der PASCAL-Syntax im Text der adressierten Einheit gesucht (einschließlich der Kommentare, da auch Kommentare auf die verwendeten Bezeichnungen Bezug nehmen). Hierdurch ist es z.B. möglich, einen Bezeichner A in einer Prozedur durch den Bezeichner ANFWERT an allen Stellen zu ersetzen, ohne daß in anderen Morphemen, die auch den Buchstaben A enthalten, eine Änderung bewirkt wird.

(Kommando: ALTERALL A ANFWERT)

Durch das Kommando

ALTERALL/[/(./

ist es möglich, alle öffnenden eckigen Klammern durch die gleichwertige Zeichenfolge '(.' zu ersetzen.

Die SEARCH-Operation ist z.B. zum schnellen Positionieren auf benannte Syntaxeinheiten geeignet:

SEARCH HILFSPROZ

würde nach dem Start des Editierens für das Programm BEISPIEL1 auf den Kopf der Prozedur HILFSPROZ positionieren.

## 9. Analyseoperationen

Als Analyseoperationen werden zunächst eine Strukturanalyse und der Syntaxtest realisiert. Die Strukturanalyse erkennt nur die Anfänge der elementaren und nichtelementaren Syntaxeinheiten (sofern sie am Anfang einer Zeile beginnen). Sie wird bei der Programmeingabe aufgerufen, um Strukturübersichten ausgeben zu können. Die Syntaxanalyse ruft der Benutzer durch das Kommando

SYNTAX OBJ1

auf, wobei OBJ1 wie bei den Änderungsoperationen definiert ist. Die Syntaxanalyse formatiert den Quelltext so, daß jede elementare und nichtelementare Syntaxeinheit am Zeilenanfang

beginnt.
Strukturanalyse und Syntaxanalyse erkennen syntaktische Fehler und geben eine Fehlerliste aus. Mit Hilfe des Kommandos

ERROR

können später fehlerhafte Einheiten adressiert werden. Es wird nach diesem Kommando in der editierten Einheit der nächste einer Syntaxeinheit oder Zeile zugeordnete Fehler gesucht. Die Ausgabe des Editors wird nach dem Kommando ERROR außerdem um eine Übersicht über die Anzahl der enthaltenen Fehler erweitert. Z.B. könnte die Ausschrift nach dem Kommando ERROR die im folgenden Beispiel angegebene Form besitzen.
Beispiel:

```
:ERROR
! BEISPIEL3**4**.1
1_  PROGRAM**0**, +++ TYPE**1**, 3_CONST**1**,..5_
                      !
  *****ERROR  2  WRONG SEQUENCE OF SYNTACTICAL UNITS
```

Das Programm BEISPIEL3 enthält vier Fehler. Davon ist ein Fehler im Konstantendefinitionsteil und ein Fehler im Variablendeklarationsteil enthalten. Ein Fehler betrifft die Reihenfolge der Syntaxeinheiten. Über den vierten Fehler erscheint aus Platzgründen keine Information.

```
:2<=3      (Vertauschen der beiden Syntaxeinheiten TYPE und
            CONST)
1_PROGRAM, +++ CONST, 3_TYPE, 4_VAR, 5_BEGIN
:ERROR     (Auf das Kommando ERROR hin wird im Konstanten-
            definitionsteil der nächste Fehler gesucht)
! BEISPIEL3**3**.CONST**1**.2_
1_  CONST
+++     K:=3;
          !
*****ERROR  10     EXPECTED
3_      MAX = 100;
```

Durch wiederholtes Anwenden des Kommandos ERROR ist es möglich, nacheinander alle Fehler zu adressieren und zu beseitigen.

## 10. Literatur

/1/ Teitelbaum, R.T.: The Cornell program synthesizer: A microcomputer implementation of PL/CS. Cornell University, Ithaca, New York, TR 79-370, 1979.

/2/ Koster, C.H. u.a.: Software development in the CDL2 laboratory. Sofware Engineering Environments, Proceedings of the GMD Symposium in Lahnstein, North-Holland, 1981, S.97-118.

/3/ Donzeau-Gauge, V. u.a.: A structure-oriented program editor: A first step towards computer assisted programming. International computing symposium, 1975, North-Holland, 1975, S.113-120.

/4/ Burkhard, H.: Konzepte zur Systematisierung der Benuzerschnittstelle in interaktiven Systemen und ihre Anwendung auf den Entwurf von Editoren. ETH Zürich, 1981.

/5/ Bothe, K.; Kosciolowicz, R.: M-PASCAL, Sprachbeschreibung. HU Berlin, Sekt. Math., Forschungsbericht, 1982.

/6/ Paulin, G.; Schiemangk, H.: Programmieren mit PASCAL. Berlin: Akademie-Verlag, 1981.

## Verfasser:

Dr. sc. B. Hohberg
Humboldt-Universität zu Berlin
Sektion Mathematik
1086 Berlin
Postfach 1297

Kosciolowicz, Raimund

# Konzeption eines Programmierlabors zur Unterstützung des modularen Programmentwurfs mit PASCAL

## 1. Entwicklungsrichtungen der Softwaretechnologie

Unter dem Eindruck der Softwarekrise kam es in den 70er Jahren zu verstärkten Forschungs- und Entwicklungsaktivitäten auf verschiedenen Teilgebieten der Softwaretechnologie. Das gemeinsame Ziel war die Entwicklung von Methoden und Hilfsmitteln zur Beherrschung der Probleme des Entwurfs, des Tests und der Wartung von Software, die Reduktion der Komplexität der Problemlösung und die Schaffung zuverlässiger und modifizierbarer Software mit Hilfe von leistungsfähigen Programmiersystemen. Diese Entwicklung hat sich in den letzten Jahren besonders auf die weitgehende Unterstützung aller Etappen des Softwarelebenszyklus durch Programmiersysteme konzentriert und findet ihren Ausdruck in einer kaum zu bewältigenden Fülle von Veröffentlichungen. In den folgenden eng miteinander verbundenen Teilgebieten der Softwaretechnologie wurden bisher bedeutende Fortschritte erzielt.

### 1.1. Entwurfsmethodik

Der gesamte Entwurfsprozeß von der Spezifikation des Problems bis zur Wartung der fertigen Lösung wurde umfassend analysiert. Dabei wurden verschiedene Ursachen für die Entstehung fehlerhafter Software aufgedeckt und neue Entwurfsmethoden entwickelt /13/. Durch die Benutzung neuer Ausdrucksmittel oder die Einschränkung der Möglichkeiten des Programmierers können viele Fehlerquellen vermieden und die Zuverlässigkeit der entstehenden Softwareprodukte erhöht werden. Dazu zählen:

- strukturierte Programmierung
- modulare Programmierung
- schrittweise Verfeinerung (Top-down-Entwurf)
- grafische Hilfsmittel für den Entwurf (z.B. Struktogramme, HIPO)

- datenorientierter Entwurf (Jacksonmethode)
- sichere Programmierung

In Abhängigkeit vom Anwendungsgebiet kommen meist mehrere Methoden kombiniert zur Anwendung. Besonders gute Ergebnisse bei der Zuverlässigkeit der entstehenden Software und der Reduktion der Entwicklungskosten wurden dort erreicht, wo statt einer streng reglementierten Anwendung der Methoden eine schöpferische Übertragung durch den Programmierer auf die konkrete Aufgabenstellung gelang /5/.

## 1.2. Programmiersprachen

Im Zusammenhang mit den Erkenntnissen der Programmiermethodik entstanden verschiedenen Varianten vorhandener Programmiersprachen (z.B. strukturierte Sprachversionen für FORTRAN und PL/1), eingeschränkte Sprachvarianten höherer Programmiersprachen für Klein- und Mikrorechner sowie neue Programmiersprachen für die universelle und fachgebietsorientierte Anwendung, die ausgehend von den Ergebnissen unter 1.1 entwikkelt wurden (z.B. PASCAL). Damit können die meisten Probleme des "Programming in the small" /6/ gut behandelt werden.

Für die Lösung komplexer Aufgabenstellungen "Programming in the large" sind vor allem Hilfsmittel zur rationellen Aufgabenzerlegung in Teilbausteine einschließlich exakter Schnittstellenbeschreibungen und -kontrollen erforderlich. In Form von Paketen und Moduln findet man diese Ausdrucksmittel in modernen Programmiersprachen wie ADA /8,9/, MODULA 2 /14/ und anderen . Zusammen mit den Möglichkeiten der separaten Übersetzung und der Anwendung des Prinzips der abstrakten Datentypen (Zusammenfassung von Daten und Operationen damit in geschützten Kapseln mit exakten Schnittstellen) bei der Programmierung ergeben sich vielfältige neue Möglichkeiten /2,10/. Wie bei jeder neuen Methode und jedem Hilfsmittel bedarf es dazu aber entsprechender Vorkenntnisse und Erfahrungen der Programmierer im Umgang damit.

Parallel dazu entstanden neue Ausdrucksmittel zur fehlertoleranten Programmierung (z.B. Assertions, Exceptions), die sowohl während des Tests als auch beim späteren Einsatz der

Software eine größere Zuverlässigkeit und Stabilität auch in abnormalen Fällen gewährleistet. In zunehmendem Maße werden auch informale und formale Spezifikationsmethoden bei der Aufgabenformulierung, dem Entwurf, dem Test und der abschließenden Dokumentation von Software angewendet. Sie ermöglichen eine exaktere Dokumentation, eine bessere Verständlichkeit und Modifizierbarkeit bis hin zu ersten gelungenen rechnergestützten Beweisen der Korrektheit von Software in speziellen Anwendungsgebieten. Besonders mit dem Vordringen der Mikroelektronik in neue Gebiete der Volkswirtschaft entstehen höhere Anforderungen bezüglich der Korrektheit und Anpaßbarkeit der dafür notwendigen Software.

## 1.3. Programmierumgebungen

Mit dem ständig steigenden Einsatz insbesondere von Klein- und Mikrorechnern in neuen Anwendungsgebieten und dem damit verbundenen starken Bedarf an spezieller Software ergab sich die Notwendigkeit der Verbesserung der rechentechnischen Unterstützung der Softwareproduktion /1/. Die in den vorhandenen Betriebssystemen enthaltenen Komponenten (Bibliotheken für Quelltexte und Programme, Editoren, Compiler, Linker, Test-Unterstützungen und Dump-Ausgaben) sowie deren fast ausschließliche Nutzungsmöglichkeit im Stapelbetrieb entsprachen nicht mehr den Wünschen der Nutzer. Erste Fortschritte wurden erreicht durch eine zusätzliche Dialognutzung vorhandener Programme(z.B. TSO) und durch die Entwicklung von Software mit Hilfe von Crossystemen auf leistungsfähigeren Rechnern. Weitere Serviceprogramme und vor allem der Einsatz von Dialogeditoren und -compilern ermöglichem dem Nutzer eine echte und flexible Dialogarbeit. Es entstanden umfangreiche Analyse-, Protokoll- und Testunterstützungen für verschiedene Programmiersprachen, bei denen während des Tests interpretierend gearbeitet wird und für den endgültigen Einsatz eine optimierende Codeerzeugung zur Verfügung steht. Das Prinzip der sequentiellen und untergliederten Dateien erfuhr eine Erweiterung durch nutzerspezifische Filesysteme mit einfachem Zugriff und entwurfsgerechter Datenverwaltung.

Eine vollständige und einheitliche Unterstützung des gesamten Entwurfsprozesses durch ein Programmiersystem von den ersten Grobvorstellungen über Entwurf, Codierung und Test bis zur Wartung der fertigen Softwareprodukte ist das Wunschziel vieler Anwender /12/. Solche Programmiersysteme erlauben aber meist nur die Verwendung einer Programmiersprache wie z.B. das CDL2-Labor /11/.

In vielen Fällen läßt sich die gestellte Aufgabe statt durch Neuprogrammierung rationeller durch Verwendung, Kopplung oder Anpassung vorhandener Programme lösen. Dies scheitert aber noch zu oft an fehlender Information oder unzureichender Dokumentation der vorhandenen Software. Abhilfe kann erst eine teilautomatisierte Sammlung, Verwaltung und Bereitstellung von Softwarelösungen in Programm-, Modell- oder Methodenbanksystemen schaffen, die die Nutzer im Dialog bei der Suche und Zusammenstellung der benötigten Teile unterstützen.

## 2. Anforderungen des Nutzers an ein Programmiersystem

Die heute noch weit verbreiteten Bedingungen bei der Herstellung von Software entsprechen in den meisten Fällen nicht den Wünschen der Nutzer. Die vorhandenen Betriebssysteme mit ihren Teilprogrammen und Datenverwaltungsmethoden erzwingen eine Anpassung der Denk- und Arbeitsweise der Nutzer an die vorhandenen Hilfsmittel, Kommandosprachen und Datenstrukturen mit dem Ziel einer möglichst effektiven Auslastung der vorhandenen Ressourcen an Speicherplatz und Rechenzeit. Dies ist unter anderem eine der Ursachen für die nicht ausreichende Qualität der entstehenden Software. Mit dem ständig steigenden Ausrüstungsgrad an moderner leistungsfähiger Rechentechnik ist aber der Zeitpunkt gekommen eine Inversion des Verhältnisses Nutzer - Rechner durchzusetzen. Ausgangspunkt müssen die Nutzerwünsche und ein entsprechender Komfort für deren Realisierung sein. Dazu gehören vor allem rationelle Dialogarbeit, leicht erlernbare Kommandosprachen, Unterstützung problemorientierter Lösungen sowie programmiersprachliche Hilfen bei Entwurf, Implementation und Test. Die internationale Entwicklung /12,13/ geht in diese Richtung und

weist als Ergebnis eine schnellere und kostengünstigere Herstellung von Software bei deutlicher Verbesserung ihrer Zuverlässigkeit nach. Moderne Programmiersysteme stellen dem Nutzer Hilfsmittel bereit für die Lösung der bei der Softwareproduktion entstehenden Teilaufgaben. Die Wirksamkeit dieser Hilfsmittel ist um so größer, je besser sie untereinander abgestimmt sind und den Nutzer von seinen Routineaufgaben entlasten /1,4/. Eine sehr wichtige Rolle spielt dabei die Organisation der Datenbasis und ihre Benutzungsmöglichkeiten.

## 2.1. Entwurfsgerechte Verwaltung, Speicherung und Adressierung der Daten

Die in den meisten Betriebssystemen übliche Datenverwaltung entsprechend den physischen Eigenschaften der Geräte und Datenträger (sequentiell, Direktzugriff, Bibliotheken) erfordert vom Nutzer einen größeren zusätzlichen Aufwand, denn sie entspricht nicht den Einheiten und Begriffen, in denen der Nutzer beim Programmentwurf und -test denkt und arbeitet. Der Nutzer verwendet Namen und logische Einheiten von Daten- und Programmstücken, die sich entweder aus dem zu lösenden Problem oder aus den Begriffen der benutzten Programmiersprache ableiten lassen (z.B. Felder, Listen, Algorithmen, Prozeduren). Er wird aber bei der heute noch weit verbreiteten Art der Datenverwaltung gezwungen, zusätzlich mit Begriffen wie Bibliothek, Bestandsname und dessen Inhalt, Zeilenstruktur, physischer Satz und ähnlichen Einheiten zu operieren und seine Programme und Daten diesen unterzuordnen. Damit ergeben sich zusätzliche Schwierigkeiten, die häufig zu Fehlern und Verzögerungen führen. Die in den Nutzerprogrammen vorhandene natürliche Hierarchie und Namensgebung (s. Abbildung 1) geht dabei weitgehend verloren, da es dazu kein Äquivalent in der Datenverwaltung gibt. Deshalb muß eine entwurfgerechte Datenverwaltung folgende Bedingungen erfüllen:

- Begriffe und Einheiten, in denen der Nutzer denkt und arbeitet, bestimmen die Struktur der verwalteten Daten
- Namen, die in Programmen und Daten vorkommen, werden zur Adressierung verwendet.

- die vorhandene natürliche Hierarchie wird bei der Speicherung und Adressierung benutzt
- Zugriffsrechte zu den Datenmengen müssen nutzerabhängig gestaltet werden
- elementare Datenmengen sollen einen überschaubaren Umfang haben, damit sie auch im Dialog bearbeitet werden können.

| Level | Quelltext |
|---|---|
| 1 | PROGRAM TRANSFORM( INPUT, OUTPUT ); |
| 2 | PROCEDURE INP( ); |
| | : |
| | : |
| 2 | END; (* INP *) |
| 2 | PROCEDURE CHECK( ... ); |
| 3 | PROCEDURE ERROR( ); |
| | : |
| | : |
| 3 | END; (* ERROR *) |
| 3 | PROCEDURE SEARCH ( ); |
| 4 | PROCEDURE APPLY ( ); |
| | : |
| | : |
| 4 | END; (* APPLY *) |
| | : |
| | : |
| 3 | END; (* SEARCH *) |
| 3 | PROCEDURE IGNORE( ); |
| | : |
| | : |
| 3 | END; (* IGNORE *) |
| | : |
| | : |
| 2 | END; (* CHECK *) |
| | : |
| | : |
| 1 | END. (* TRANSFORM *) |

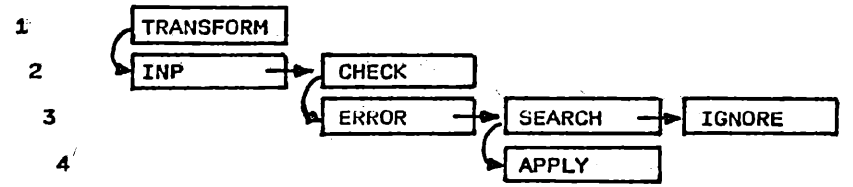



Abbildung 1: Natürliche Einheiten und Hierarchie in einem Nutzerprogramm

Aus den obigen Bedingungen folgt, daß es notwendig ist, die Datenverwaltung mit Hilfe von Strukturbäumen zu organisieren, deren Aufbau sich unmittelbar aus dem Daten ableiten läßt und deren Feinheitsgrad und Knoteninhalte der Nutzer selbständig bestimmen kann. Alle Veränderungen, die während des Softwareentwurfs vorgenommen werden, lassen sich mit Hilfe von Operationen an diesen Strukturbäumen durchführen /3/.

## 2.2. Unterstützung verschiedener Arbeitsweisen der Nutzer

Leistungsfähige Programmiersysteme zwingen den Nutzer nicht zu einer bestimmten Arbeitsweise, z.B. strenger Top-down-Entwurf, sondern unterstützen verschiedene Vorgehensweisen, informieren den Nutzer über Struktur und Inhalt der gespeicherten Daten und beraten ihn über mögliche weitere Arbeitsschritte (Fehlerkorrektur, Neuübersetzung, Schnittstellenprüfung). Sie ermöglichen sowohl den Entwurf neuer Programme nach top-down-, bottom-up-, funktionsorientierten, datenorientierten oder gemischten Strategien als auch die Lösung von Aufgaben durch Kombination vorhandener Routinen in Form einer Programm- oder Methodenbank /12/.

Wesentliche, von der verwendeten Nutzerprogrammiersprache unabhängige, Teile eines Programmiersystems müßten sein:

- leistungsfähige und einfach handhabbare Kommandosprache zur Gestaltung eines rationellen Nutzerdialoges
- Informationen über Aufbau und Inhalt der in den Strukturbäumen abgespeicherten Daten
- Operationen zur Bewegung in den Strukturbäumen durch relative oder absolute Angabe des Ziels verbunden mit einer Aktualisierung der erreichbaren Teile nach dem Focusprinzip
- Veränderungen der Strukturbäume durch Hinzufügen, Streichen oder Umhängen von Knoten mittels eines Baumeditors
- Veränderungen elementarer Datenknoten der Strukturbäume mittels eines Texteditors
- einfacher Datenaustausch mit der Betriebssystemumgebung

Diese Aufzählung zeigt, daß eine große Zahl der Teilaufgaben eines Programmiersystems unabhängig von der konkreten

Programmiersprache der Nutzer gehalten werden kann. Sie sind gleichermaßen einsetzbar für Sprachen wie Assembler, Fortran, PASCAL oder MODULA 2 aber auch in einem Textsystem für die Herstellung von Dokumentationen. Für jeden konkreten Anwendungsfall ist lediglich die Bedeutung der Namen und Knoteninhalte verschieden, nicht aber die Arbeit mit den Strukturen. Insbesondere in den ersten kreativen Etappen des Softwareentwurfs arbeitet der Nutzer mit verbalen oder formalen Spezifikation der Aufgabe und des Lösungsweges sowie längere Zeit mit unvollständigen meist syntaktisch fehlerhaften Programmen. Gerade dabei kann der Dialog in Form von Abspeicherung, Strukturinformation und Dokumentation wesentlich helfen, da dem Nutzer je nach Wunsch einen Überblick über das Gesamtproblem oder die Konzentration auf eine Teilaufgabe ermöglicht wird. Die im folgenden Abschnitt aufgeführten Teilaufgaben sind beliebig ab- oder aufrüstbar und bestimmen in Abhängigkeit vom Anwendungszweck den Komfort eines Programmiersystems.

## 2.3. Sprachabhängige Teilaufgaben

Zur besseren Unterstützung der Arbeit mit einer konkreten Programmiersprache insbesondere bei der Beseitigung syntaktischer und semantischer Fehler sind folgende Bestandteile in einem Programmiersystem sinnvoll:

- Analyseprogramme zur Bestimmung der Grobstruktur und Hierarchie einer sequentiellen Datenmenge als Hilfsmittel zur Aufspaltung eines Knotens verbunden mit einer vertikalen Verlängerung des Strukturbaumes entsprechen den Wünschen der Nutzer
- Aufbereitungsprogramme für die Verwendung der Daten eines Teilbaumes zu Dokumentations-, Compilations- oder Testzwecken (Schrumpfen eines Teilbaumes)
- entwurfsunterstützende Syntaxanalyse, die kontextfrei und mit vom Nutzer wählbarer Schärfe arbeiten kann
- syntaxgesteuerte Editierung von elementaren Datenknoten im Dialog mit dem Nutzer
- lokale und globale Kontextkontrollen

- Kontrolle der Schnittstellen zwischen Moduln und deren Ausgabe an den Nutzer in übersichtlicher Form
- Testvorbereitung und Testhilfen
- Übersetzung, Interpretation bzw. Abarbeitung von selbständigen Einheiten

Mit ihrer weiteren theoretischen Durchdringung und Überführung zur praktischen Nutzbarkeit werden Hilfsmittel aus folgenden Gebieten zu wertvollen Bestandteilen von Programmiersystemen:

- Messungen des Programmverhaltens und deren automatisierte Auswertung (Tuning)
- Analyse der Programme zur Bestimmung oder Abschätzung zu erwartender Eigenschaften
- Ableitung von Simulationsmodellen zur Erprobung des möglichen Programmverhaltens
- Anwendung formaler Spezifikationsmethoden zur exakten Dokumentation und für formale Beweismethoden

Die Zusammenstellung dieser Möglichkeiten zur rechentechnischen Unterstützung der Softwareproduktion zeigt das aus heutiger Sicht Realisierbare. Viele bekannte Programmiersysteme /12/ enthalten nur einen Teil davon und sind auf einen konkreten Anwenderkreis zugeschnitten. Die Masse der verfügbaren Programmierhilfsmittel entspricht aber nicht einmal den hier aufgestellten Mindestanforderungen einer entwurfsgerechten Datenverwaltung. Es erfordert noch große Anstrengungen, die Mehrzahl der bekannten Methoden und Hilfsmittel einer großen Zahl von Nutzern verfügbar zu machen.

## 3. Zielstellungen unseres Programmierlabors

Ein wichtiger Schritt auf dem Weg zu einer umfassenden rechentechnischen Unterstützung der Softwareproduktion ist die Sammlung eigener Erfahrungen bei der Entwicklung und Nutzung von Programmiersystemen unter Einbeziehung der Prinzipien und Ergebnisse bereits existierender Systeme. Die daraus resultierenden Erkenntnisse können einerseits zu verallgemeinerten Merkmalen und Richtlinien entwickelt und andererseits

bezüglich ihrer Realisierbarkeit, Effektivität und Nützlichkeit unter konkreten Einsatzbedingungen (Hardware, Software, Anwenderspektrum) überprüft werden. Dabei kann auf langjährige Erfahrungen bei der Entwicklung von Programmen in verschiedenen Betriebssystemen DOS, OS, MS mit unterschiedlichem Komfort aufgebaut werden. Gleichzeitig ist das Programmierlabor ein wirksames Hilfsmittel für Entwurf, Codierung, Test und Analyse der Programme zur Lösung von Aufgaben an unserem Bereich (Modellbetriebssysteme, Simulation, Ausbildungsaufgaben). Das Anwendungsspektrum erstreckt sich dabei von kleinen und mittleren Programmieraufgaben über die Nutzung als Programm- oder Methodenbank bis hin zur Lösung komplexer Aufgaben. Es erleichtert die Arbeit des Nutzers durch Übernahme von Arbeitsanteilen (Speicherung, Verwaltung, Analyse), Unterstützung einer rationellen Arbeitsweise mit weniger Fehlern und Information über aktuelle Zustände und mögliche Weiterarbeit im Dialog. In der ersten Variante werden vorrangig die sprachunabhängigen Teilaufgaben und einige auf PASCAL zugeschnittene sprachabhängige Teile realisiert.

### 3.1. Anwendung des Modulprinzips und der abstrakten Datentypen

Für die Entwicklung komplexer Programmsysteme bieten moderne Programmiersprachen in Form von Moduln in MODULA 2 oder Paketen in ADA sehr leistungsfähige Ausdrucksmittel /9,10,14/. Sie gestatten eine Zerlegung größerer Aufgaben in relativ selbständige Teilaufgaben (Bausteine) einschließlich einer exakten Schnittstellenspezifikation, die statisch überprüfbar ist und eine größere Sicherheit der entstehenden Software gewährleistet. Damit ist auch das Prinzip der abstrakten Datentypen im vollen Umfang realisierbar, was bei richtiger Anwendung zu einer deutlichen Reduktion der Komplexität der Probleme sowie besserer Verständlichkeit und Modifizierbarkeit der Programme führt /2,7/.

Bei Entwurf, Implementation und Dokumentation des Programmierlabors wird das Prinzip der Strukturierung mit Moduln

im Sinne von MODULA 2 (nicht zu verwechseln mit dem weitverbreiteten und sehr unterschiedlich benutzten Begriff der modularen Programmierung) angewendet und erprobt. Unter den Bedingungen der Arbeit in kleinen Kollektiven zusammen mit Studenten ist dies ein wesentlicher Faktor zur Gewährleistung der Zuverlässigkeit der entstehenden Software. Die Realisierung des Programmierlabors erfolgt aus Gründen der besseren Portabilität und Modifizierbarkeit in einer höheren Programmiersprache. Einige wenige, für die Kopplung mit dem realen Betriebssystem zuständige Teile sind mit Hilfe der Assemblersprache zu realisieren und werden als externe Routinen aufgerufen. Da keine Compiler für ADA bzw. MODULA 2 auf den vorhandenen Rechenanlagen zur Verfügung stehen, erfolgt die Realisierung mit Hilfe einer erweiterten Sprachversion M-PASCAL von PASCAL um wesentliche Teile des Modulkonzeptes aus MODULA 2. Ein Compiler und ein Precompiler (M-PASCAL PASCAL) sind als wesentliche Implementationshilfsmittel vorhanden. Eine separate Übersetzung von Modulen konnte nicht realisiert werden, da dazu ein wesentlicher Eingriff in die Codegenerierung und in das Laufzeitsystem erforderlich wäre. Trotzdem ergeben sich durch die Strukturierung mit Moduln und die Verwendung externer Routinen bedeutende Vorteile gegenüber einer Implementation mit PASCAL.

### 3.2. Schichtenmodell des Programmierlabors

Dem Nutzer stehen in der ersten Ausbaustufe des Programmierlabors als Programmiersprachen PASCAL und M-PASCAL zur Verfügung. Eine Erweiterung auf andere Programmiersprachen ist aber wegen der allgemeineren Orientierung des Gesamtsystems jederzeit möglich. Dazu sind einige der unter 2.3. angegebenen sprachabhängigen Teile entsprechend zu erweitern. Die Abbildung 2 zeigt ein Schichtenmodell des Programmierlabors. Die Schnittstellenschicht 3 zum Wirtsbetriebssystem einschließlich der dazu notwendigen Assemblerroutinen kann dabei einfach und sicher gestaltet werden, wenn sie als abstrakter Datentyp mit exakter Spezifikation gestaltet wird, deren Operation auf einem relativ niedrigen Niveau liegen

und deren Daten nur über den Aufruf dieser Operationen mit einfacher Parametergestaltung erreichbar sind. Alle anderen Teile können mit Hilfe von M-PASCAL oder PASCAL entworfen und implementiert werden.

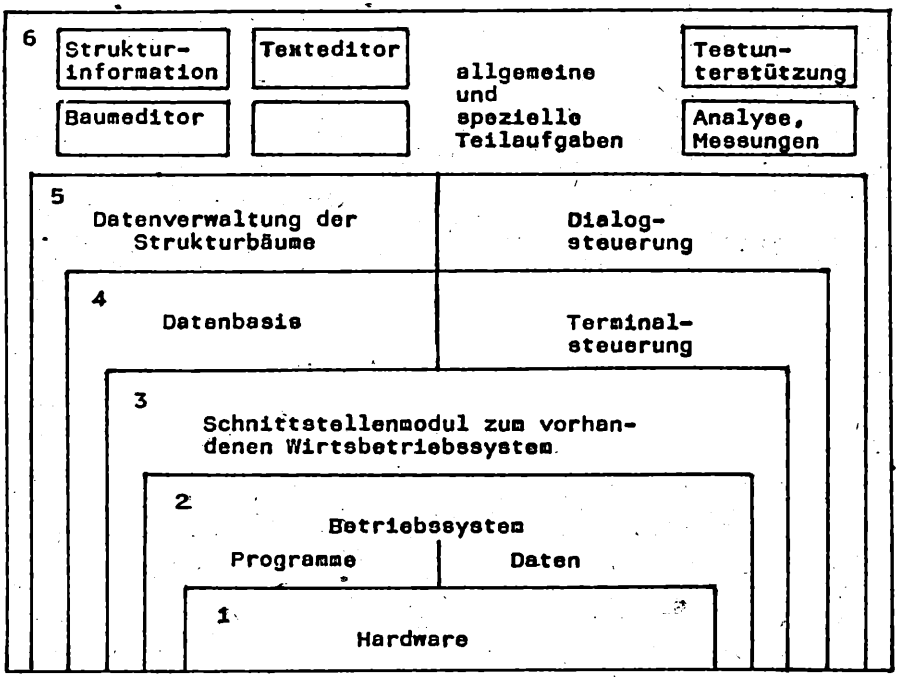



Abbildung 2: Schichtenmodell des Programmierlabors

Die Organisation einer eigenen Datenbasis (Schicht 4) unabhängig von der im Wirtsbetriebssystem vorhandenen Datenorganisation ergibt sich aus der Notwendigkeit eines leistungsfähigen nutzerorientierten Datenschutzes und den Möglichkeiten der Portabilität des Systems. Entsprechend der Leistungsfähigkeit der auf dem Wirtssystem vorhandenen Datenverwaltung kann der Umfang der Schichten 3 und 4 variiert werden. Die Probleme der nutzergerechten Datenverwaltung und der Dialogführung der Schicht 5 bestimmen wesentlich das Verhältnis Nutzer - Programmiersystem. Der erweiterbare Umfang der Schicht 6 beeinflußt die Einsatzmöglichkeiten und den Komfort

des Programmierlabors. Neben einigen sprachunabhängigen Teilen (Strukturinformation, Editoren) enthält diese Schicht alle sprachabhängigen Teile. Die Realisierung vieler Komponenten erfolgt im Rahmen der Forschungsarbeiten von Studenten und Aspiranten.

Literatur:

/1/ Baumbach, H.-D., Gühne, W. "Systeme der Rechenunterstützung in der Softwareentwicklung", Rechentechnik- Datenverarbeitung 1980 Nr. 10 S. 13-16

/2/ Bothe, K. "Restructuring a compiler by abstract data types - an experiment in using data abstractions for software modularisation", Seminarbericht Nr. 40, Humboldt-Universität zu Berlin, 1981

/3/ Burkhardt, M. "Konzepte zur Systematisierung der Benutzerschnittstellen in interaktiven Systemen und ihre Anwendung auf den Entwurf von Editoren", Institut für Informatik der ETH Zürich Bericht 43, 1981

/4/ Buxton, J. N., Druffel, L. E. "Requirements for an ADA programming support environment: Rational for Stoneman", in /12/ S. 319-330

/5/ Denert, E. "Software-Modularisierung", Informatik-Spektrum, 1979 Nr. 2 S. 204-218

/6/ De Remer, F., Kron, H. "Programming-in-the-large versus Programming-in-the-small", ACM SIGPLAN NOTICES 10(1975) Nr. 6 S. 114-121

/7/ Hohberg, B. "Abstrakte Datentypen beim Programmentwurf am Beispiel eines syntaxorientierten Programmeditors für PASCAL", WBZ TU Dresden 1982

/8/ Ichbiah, J. D. u.a. "Rational for the design of the ADA programming language", ACM SIGPLAN NOTICES 14(1979) Nr. 6 Teil B

/9/ Ichbiah, J. D. u.a. "Preliminary ADA reference manual", ACM SIGPLAN NOTICES 14(1979) Nr. 6 Teil A

/10/ Kosciolowicz, R. "Unterstützung des Programmentwurfs in der Programmiersprache ADA", WBZ der TU Dresden, Heft 47 1981

/11/ Koster, C. H. A. u.a. "Software development in the CDL2 laboratory", in /12/ S. 97-118

/12/ Software Engineering Environments, Proceedings of the GMD Symposium in Lahnstein, North-Holland, 1981

/13/ Tutorial on Software Design Techniques, third edition, IEEE Computing Society, 1980

/14/ Wirth, N. "MODULA 2", Institut für Informatik der ETH Zürich, Bericht 36, 1980

Verfasser:

Dr. rer.nat. R. Kosciolowicz
Humboldt-Universität zu Berlin
Sektion Mathematik
1086 Berlin
Postfach 1297

Horn, Christian

# Breitbandsprachen und Programmentwicklung

> Most problems have either many answers or no answer. Only a few problems have a single answer.
>
> E.C.Berkeley[1)]

## 1. Einleitung

Am Anfang steht das Wort - bei der Programmentwicklung die meist verschwommene Beschreibung der Aufgabenstellung in einer Sprache, die sich eigentlich nur "eingefleischte" Programmierer und vielleicht Mathematiker trauen als natürliche Sprache zu bezeichnen. Ein Programm durchläuft dann im Zuge seiner Entwicklung mehrere sprachliche Ebenen, über die erste präzise Spezifikation, verschiedene Stufen der applikativen (funktionalen) und imperativen (prozeduralen) Programmbeschreibung, den übersetzungsfähigen Quelltext bis hin zum ausführbaren Maschinenkode. Jede dieser Sprachen hat eine eigenständige Bedeutung: Sie widerspiegeln die für die jeweilige Entwicklungsphase des Programms relevanten Aspekte besonders deutlich und sind daher unter dem Gesichtspunkt der primär kommunikativen Funktionen einer Sprache für diese Entwurfsetappen prädestiniert. Legt man ein Phasenmodell für die Softwareproduktion zugrunde /15, 22/, so entsprechen diese sprachlichen Ebenen in etwa den Abschlußdokumenten der einzelnen Phasen; auch die Programmdokumentation setzt sich aus Einzeldokumenten verschiedener dieser Sprachen zusammen.

Die Verwendung mehrerer sprachlicher Ebenen birgt jedoch Gefahren. Die Übergänge zwischen den einzelnen Sprachen schaffen zusätzliche Unsicherheiten im Entwurfsprozeß. Sie erfordern besonders große Sorgfalt, wenn es gilt, das Zusammenspiel von Programmkomponenten zu realisieren, die jeweils unterschiedliche Entwicklungsstadien repräsentieren.

---

1) zitiert nach /6/

Breitbandsprachen verkörpern nun einen Kompromiß. Sie versuchen, den gesamten Bereich von der Problemspezifikation bis zum direkt übersetzungsfähigen und genügend effektiven Quellprogramm in einer Sprache zu überdecken. Sie sind damit zwar auf jeder einzelnen sprachlichen Ebene den konventionellen Ausdrucksmitteln (geringfügig?) unterlegen, bieten aber größere Sicherheit bei den Übergängen zwischen den einzelnen Phasen und erlauben vor allem eine durchgängige Rechnerunterstützung über alle Phasen des Programmentwurfs. Breitbandsprachen (engl. wide-spectrum languages) müssen daher eine Vielzahl aufeinander abgestimmter Ausdrucksmittel umfassen: Für jede Klasse von Programmelementen (Datentypen, Ausdrücken und Anweisungen), die verschiedenen Entwicklungsstufen (deskriptiv, applikativ und imperativ) und die problemabhängigen Abstraktionsebenen müssen sich geeignete Darstellungsformen finden. Die Sprache muß aber zugleich übersichtlich, erlernbar und praktisch handhabbar sein.

Der vorliegende Beitrag will eine Einführung in die Struktur und den Gebrauch von Breitbandsprachen geben, wobei dem Kalkül der Programmtransformationen naturgemäß besonderes Gewicht zukommt.

## 2. Zur Organisation der Programmentwicklung

Der Programmentwurf gliedert sich zeitlich in eine Folge von Entwurfsschritten

$$(p_0, p_1, p_2, \ldots, p_n)$$

wobei jeder Entwurfsschritt $s_i: p_i \rightarrow p_{i+1}$ in der streng lokalen Anwendung einiger weniger Entwurfsoperationen besteht. Der Programmentwurf beginnt mit einer ersten Spezifikation der Aufgabenstellung ($p_0$) und endet mit dem "fertigen", direkt übersetzungsfähigen Programm ($p_n$). Am Anfang steht jedoch nie eine vollständige Spezifikation der Aufgabenstellung. Erst im Zuge der Programmentwicklung kommt es zu einer schrittweisen Annäherung der Spezifikation an die eigentliche Aufgabenstellung, wobei die Spezifikation dann bereits derartig über das Programm "verteilt" ist, daß es kaum noch

möglich ist, von der Spezifikation des fertigen Programms zu sprechen. Weiterhin ist eine rein deskriptive Beschreibung realer Programme wenig nutzbringend weil unnötig kompliziert. Viele Prozesse und Eigenschaften sind in ihrer algorithmischen Darstellung leichter faßlich.

Die Korrektheit des Programms $p_n$ bezüglich der Anfangsspezifikation $p_0$ ist in den meisten Fällen trivial erfüllt. Wir wollen daher die Korrektheit des Entwurfsprozesses einführen und definieren:

Def. Ein Entwurfsprozess $(p_0, p_1, \ldots, p_n)$ heißt korrekt, wenn für beliebige $i,j$ mit $0 \leq i < j \leq n$ das Programm $p_j$ bezüglich $p_i$ korrekt ist. Ein Entwurfsschritt $s_i: p_i \rightarrow p_{i+1}$ heißt korrekt, wenn $p_{i+1}$ bezüglich $p_i$ korrekt ist.

Jedes vorangehende Entwurfsstadium eines Programms muß also in gewissem Sinne als "Spezifikation" eines nachfolgenden Entwurfsstadiums angesehen werden. Es ist notwendig, die Korrektheit, bisher als Relation zwischen rein deskriptiver und imperativer Darstellung eines Programms betrachtet, weitgehend zu verallgemeinern. Grundlage hierfür sind die Arbeiten zur Äquivalenz und Einbettung von Programmen /7/. Durch Induktion sieht man sofort, daß ein Entwurfsprozess genau dann korrekt ist, wenn jeder der konstituenten Entwurfsschritte korrekt ist. Auf das Problem der Korrektheit von Entwurfsschritten kommen wir im nächsten Abschnitt zurück.

Die einzelnen Entwurfsschritte sind von sehr unterschiedlichem Charakter: einige verkörpern fundamentale Entwurfsentscheidungen, andere schöpfen einen durch nichtdeterministische bzw. rein deskriptive Konstruktionen gegebenen Spielraum aus. Die Mehrzahl der Entwurfsschritte besteht jedoch in der Anwendung wohlbekannter Entwurfsoperationen auf konkret vorgegebene Teilstrukturen. In den meisten Fällen besteht die Auswahl nur zwischen wenigen alternativen Lösungsvarianten, jeder Programmierer hat sich aufgrund seiner Erfahrungen gewisse "Standardlösungen" zu eigen gemacht. Längst bekanntes wird in monotoner Weise ständig wiederholt. Trotzdem steckt die Rechnerunterstützung für diese Tätigkeiten noch in den Kinderschuhen /16/.

Die gegenwärtige Technologie der Softwareproduktion wird durch drei wesentliche Elemente geprägt: die Entwurfsmethode, die sprachliche Basis und die verwendeten technischen Hilfsmittel (Werkzeuge).

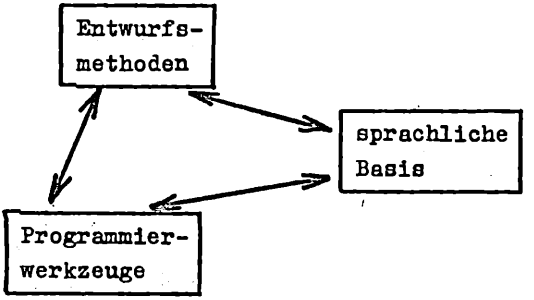



Abb.1 Elemente der Softwaretechnologie

Die verschiedenen Entwurfsmethoden lassen sich nach globalen und lokalen Merkmalen unterscheiden. Geht man von dem allgemein üblichen Baummodell für die Programmstruktur aus, so durchläuft jeder Knoten (jedes Programmelement) im Zuge der Programmentwicklung (möglicherweise in geraffter Form) die einzelnen Entwurfsetappen. Die Wechselwirkung zwischen Programmelementen beschränkt sich dabei in der Regel auf benachbarte Knoten (Lokalitätsprinzip). Globale Merkmale beschreiben die Reihenfolge, in der die einzelnen Knoten der Baumstruktur nacheinander eine gewisse vorgegebene Entwicklungsstufe durchlaufen (vgl. Abb.2). Lokale Merkmale beschreiben die Folge der Generationen eines einzelnen Programmelements von der ersten Spezifikation auf relativ hohem Abstraktionsniveau (Problemebene) bis hin zur Implementation der präzisierten Aufgabenstellung auf vergleichsweise niedrigem (Maschinen-)Niveau. Zur Darstellung dieser Generationenfolge haben sich Entwurfsdiagramme (engl. refinement-realization-space /10/) bewährt (vgl. Abb.3).

Natürlich erlaubt diese Klassifikation der Entwurfsmethoden (vgl. Abb.4) nur eine grobe Einteilung: dazwischen gibt es eine Vielzahl von Abstufungen. Praktisch eingesetzte Methoden

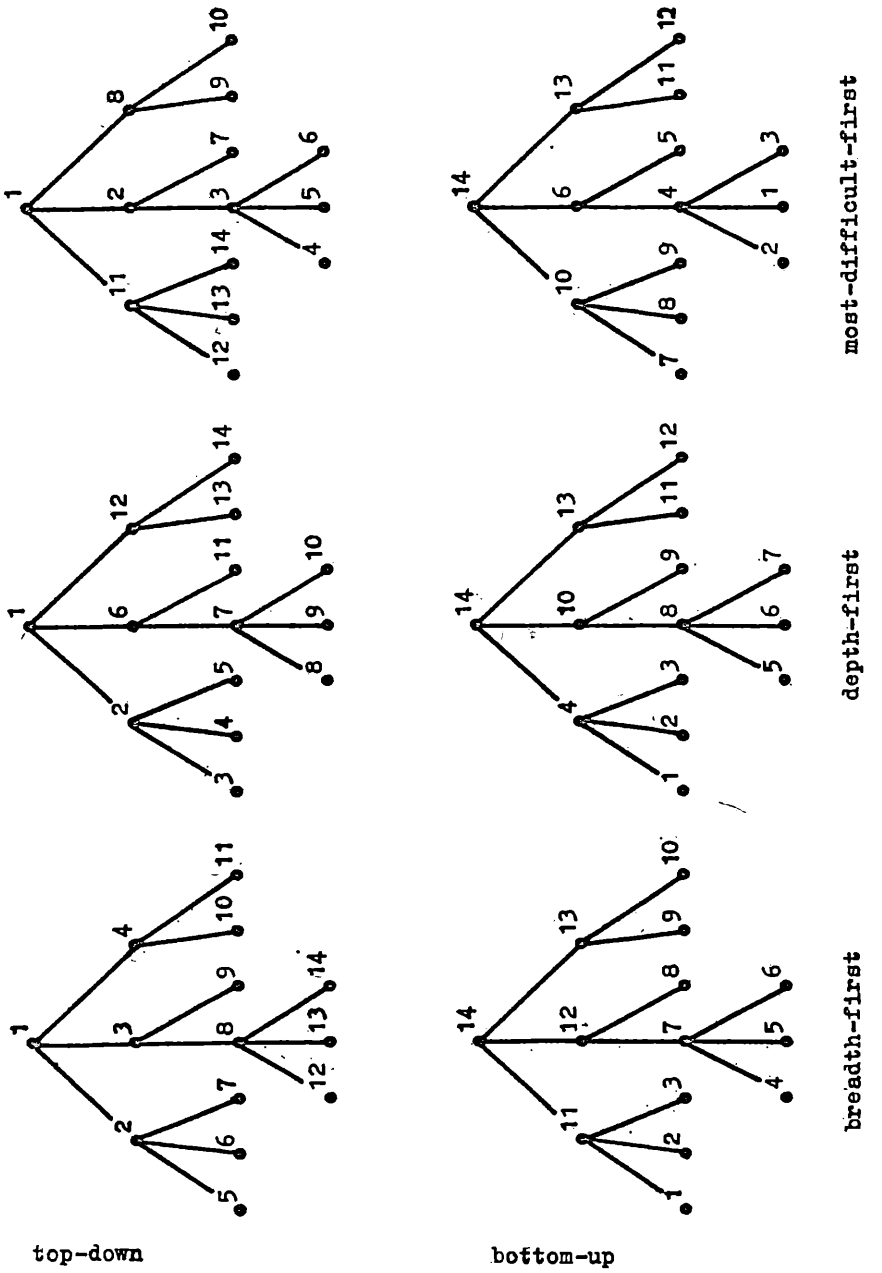



Abb.2 Globale Organisation des Programmentwurfs

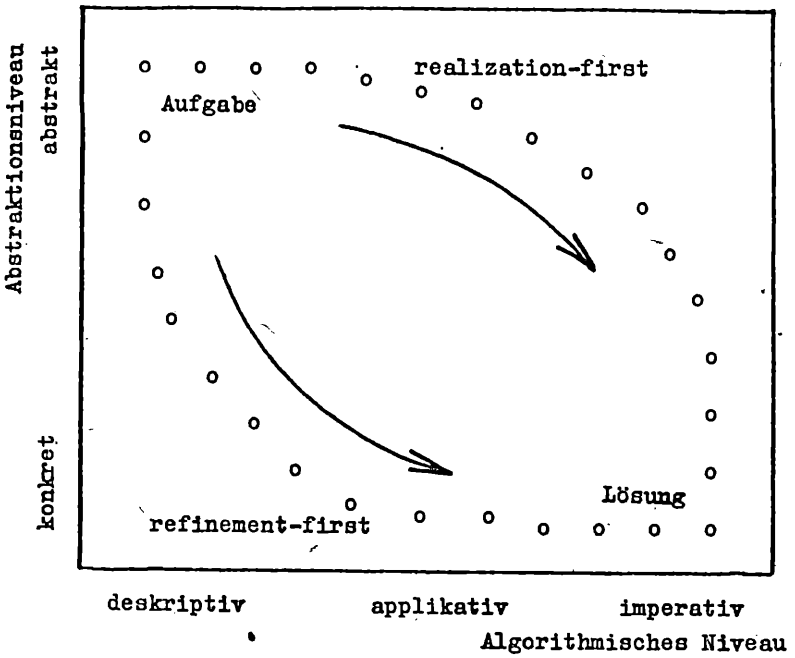



Abb.3 Lokale Organisation des Programmentwurfs

stellen immer eine Verquickung der "abstrakten Methode" mit speziellen sprachlichen Mitteln und Werkzeugen dar. Glaubte man in der Anfangszeit des "Software-Engineering", durch eine geeignete, festgefügte Organisation der Programmentwicklung das Softwareproblem bewältigen zu können, so wissen wir heute,

- Entwurfsstrategien
  - top-down
  - bottom-up
- Entwurfsprinzipien
  - breadth-first
  - depth-first
  - most-difficult-first
- Entwurfstechniken
  - refinement-first
  - realization-first

Abb.4 Klassifikation der Entwurfsmethoden

daß eine Methode allein nur in engen Spezialfällen zum Erfolg führen kann. Die Zeit hitziger Diskussionen um einzelne Methoden ist im wesentlichen vorbei. Wir kennen annähernd die Wirkbedingungen, unter denen die verschiedenen Methoden nutzbringend angewandt werden können - auch wenn diese Erkenntnisse noch nicht im notwendigen Maße praxiswirksam werden. Der Programmierer beginnt in der Wahl seiner Mittel freier zu werden - freier im dialektischen Sinne: er beginnt die zur Lösung der vorgegebenen Aufgabe geeignetsten Mittel und Methoden bewußt einzusetzen. Der Trend geht zu für die praktische Anwendung aufbereiteten Bündeln von Methoden, aufeinander abgestimmten sprachlichen Mitteln und Softwarewerkzeugen /15/. Doch steht dem häufig die Unverträglichkeit der verschiedenen sprachlichen Ausdrucksmittel entgegen.

Breitbandsprachen verkörpern ein verallgemeinerndes, einheitliches aber erweiterbares Sprachkonzept, das die Integration verschiedener Entwurfsmethoden unterstützt/5,6/. Es kann sich nicht darum drehen, eine Universalsprache zu schaffen, die es erlaubt, alle Probleme mit einem Schlage zu lösen. Breitbandsprachen stellen zunächst ein Konzept dar - genau wie die geläufigen modernen Programmiersprachen bei all ihren Unterschieden dem Konzept der prozeduralen algorithmischen Sprache folgen. Neue Sprachen setzen sich nicht im luftleeren Raum durch. Es wird daher ganz natürlich sowohl PASCAL- als auch ALGOL-, PL/1- oder Ada-ähnliche Breitbandsprachen geben. Ansätze hierzu sind die PASCAL- und ALGOL-Versionen von CIP-L /3-6, 21, 25/ und ABSTRACT-PASCAL/13,14,18/. Für einige Anwendungsgebiete bieten sich auch fachsprachliche Erweiterungen bzw. Versionen dieser universellen Breitbandsprachen an. Wenn dadurch erreicht würde, daß formale Spezifikationen für den Fachwissenschaftler oder Ingenieur als natürliche Ausdrucksweise ihres Anliegens akzeptiert werden, dann hätten wir einen wesentlichen Schritt vorwärts getan.

## 3. Rechnerunterstützung für den Programmentwurf

Im Laufe der Jahre hat sich eine relativ feste Dateiorganisationsform für die Programmentwicklung herausgebildet, die von der Mehrzahl der im Einsatz bzw. in der Entwicklung

befindlichen Programmiersysteme genutzt wird /13, 14, 17, 18, 20, 24/. Das "Skelett" dieser Dateien bildet eine Baumstruktur, deren Knoten relativ selbständige Programmelemente darstellen. Jeder Knoten beinhaltet verschiedene Generationen und Versionen ein und desselben Programmelements. Die Verkettung der Knoten untereinander widerspiegelt die Block- bzw. Modulstruktur des Gesamtprogramms. Die Adressierung erfolgt hierarchisch über die Namen der Programmelemente. Es gibt keine speziellen "Bestandsnamen". Eine Vereinfachung der Adressierung unter Berücksichtigung der Sichtbarkeitsregeln für Bezeichner ist möglich. Für die einzelnen Knoten

```
program P
  module storage
    type address = ...
    var s :
    proc put, get
  begin "initialisierung" end
  proc input, process, output ...
begin "hauptprogramm" end.
```

Beispielprogramm

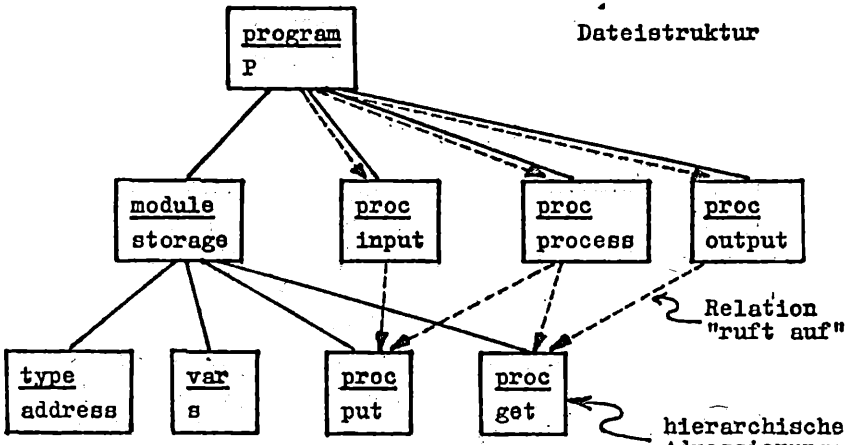



Abb.5 Dateistruktur

sind eine Vielzahl von Prädikaten (z.B. "ist syntaktisch fehlerfrei/fehlerhaft", "ist bereits übersetzt" oder "ist erfolgreich verifiziert") und Relationen (z.B. "ruft auf", "benutzt", "ist unverträglich mit") definiert, deren Verwaltung sich auf Standarddatenbankmechanismen stützt /16,20/. Abb.5 zeigt die Dateistruktur für ein kleines Beispielprogramm.

Beim Entwurf von Programmen werden immer wieder die gleichen elementaren Operationen ausgeführt. Egal ob dazu ein syntax- oder zeichenorientierter, ein muster- oder adressengesteuerter Editor oder ein anderes leistungsfähiges Programmierwerkzeug benutzt wird, für den Programmierer bleibt ein großer Teil monotoner, mechanischer Arbeit. Moderne Programmiersysteme besitzen daher eine eigene (an die zugrundeliegende Programmiersprache bzw. an das Basisbetriebssystem angeglichene) Steuersprache /17,24/, die es gestattet immer wiederkehrende Folgen von Entwurfskommandos zu Entwurfs"prozeduren" zusammenzufassen. Es wird ein Sortiment von Grundkommandos bereitgestellt. Mittels bedingter, rekursiver und iterativer Konstruktionen lassen sich über Prädikate und Relationen gesteuert komplexe Entwurfsabläufe vorprogrammieren. Man spricht hierbei bereits von "Metaprogrammierung" /11/.

Die entscheidende Frage liegt nun darin, welche Grundkommandos für den Programmentwurf zur Verfügung gestellt werden. Die "normalen" Updatefunktionen (Einfügen, Streichen oder Modifizieren von Zeichenketten oder Syntaxeinheiten) sind als Grundfunktionen in intelligenten Terminals realisiert und stehen damit jederzeit zur Verfügung. Nach derartigen Änderungen ist jedoch stets die Konsistenz des modifizierten Programmelements mit seiner Umgebung zu überprüfen. Da wir von vornherein an korrekten Entwurfsschritten interessiert sind, wäre es ratsam als Grundoperationen korrektheitserhaltende Programmtransformationen aufzunehmen. Eine beliebige Verkettung solcher "Mikrotransformationen" bildet wiederum eine korrekte Programmtransformation, u.U. mehrere solcher Programmtransformationen bilden einen korrekten Entwurfsschritt. Das erste Programmentwicklungssystem auf der Basis von Programmtransformationen stammt von Burstall und Darlington /8/. Inzwischen sind in der Literatur über ein halbes Dutzend derartiger Systeme belegt /4, 19/.

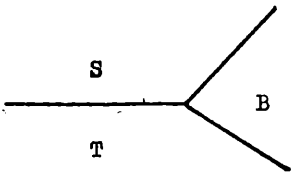
## 4. Programmtransformationen

Alle Methoden für den (schrittweisen) Entwurf korrekter Programme stützen sich auf verschiedene Formen semantischen Schließens. Werden jedoch mechanische Systeme zur Unterstützung des Programmentwurfs und zur Absicherung der Korrektheit der einzelnen Entwurfsschritte eingesetzt, so müssen diese semantischen Beziehungen syntaktisch charakterisiert werden. Der Übergang von einer Generation eines Programmelements zur nächsten wird am einfachsten durch formale Transformationsregeln beschrieben, ähnlich den Schlußregeln der formalen Logik.

Allgemein gesprochen ist eine Programmtransformation R eine partielle Abbildung, die einem Programmstück S der Eingangssprache $L_1$, sofern die Anwendbarkeitsbedingung B erfüllt ist, ein neues Programmstück T=R(S) der Ausgangssprache $L_2$ zuordnet. Die Vorteile des transformationellen Ansatzes, insbesondere die weitgehende Rechnerunterstützung, die sich aus der Möglichkeit beliebiger Verkettung von Transformationen ergibt, können in den meisten Fällen jedoch nur dann genutzt werden, wenn Eingangs- und Ausgangssprache zusammenfallen, wenn $L_1=L_2$. Zur Darstellung von Programmtransformationen gibt es prinzipiell zwei Möglichkeiten:

(i) Eine Programmtransformation kann in Form eines Algorithmus gegeben werden, der die Abbildung R realisiert, d.h. auf die Eingabe eines Programmstückes S mit der Ausgabe des Programmstückes T=R(S) antwortet. Compiler arbeiten z.B. nach diesem Schema.

(ii) Eine Programmtransformation kann aber auch als geordnetes Paar von Programmschemata dargestellt werden, bestehend aus einem "Eingabemuster" S und einem "Ausgabemuster" T /19, 23/, die der Prämisse bzw. Conclusio einer Postschen Regel entsprechen, und ggf. einer Anwendbarkeitsbedingung B. Die Anwendung einer Programmtransformation auf ein gegebenes Programm ist dem Substitutionsschritt eines Postschen Systems vergleichbar.




.Abb.6 Schema für Transformationsregeln.

Es gibt verschiedene Ansätze, die für den Programmentwuf zweckmäßigen Programmtransformationen in geeigneter Weise zu systematisieren. Dabei zeichnen sich zunächst zwei Ebenen ab: Basisregeln, die die kleinsten durch Transformationen lösbaren Aufgaben beschreiben, und sogenannte abgeleitete Regeln, die sich in wohl definierter Weise aus Basisregeln zusammensetzen. Als System von Basisregeln hat wahrscheinlich ein Regelsystem von Gentzen'schen Typ die größten Aussichten: Ein solches Regelsystem enthält nicht nur für jeden logischen Funktor, sondern darüberhinaus für jede Operation, für jede Funktion und jede algorithmische Konstruktion eine Einführungs- und eine Eliminationsregel.(vgl. Abb.7). Die bekannten FOLD- und UNFOLD-Regeln sind hier als Einführungs- bzw. Eliminationsregel für Funktionsaufrufe einzuordnen.

Für jede der Transformationsregeln muß natürlich die Korrektheit bewiesen werden. Eine Programmtransformation heißt korrekt, wenn es eine semantische Einbettung von T in S gibt, d.h. wenn zu jeder Interpretation von T eine schwach (d.h. in bezug auf das äußere Verhalten) äquivalente Interpretation von S existiert. Für den Nachweis der Korrektheit einer Programmtransformation gibt es wiederum zwei grundsätzliche Möglichkeiten. Zum einen kann man die obige Eigenschaft unter Bezugnahme auf die Semantik der Sprachen $L_1$ und $L_2$ direkt beweisen. Man kann aber auch eine Anzahl von Basisregeln als gegeben (Axiome) betrachten, und versuchen, neue Transformationen konstruktiv als Nacheinanderausführung dieser Basisregeln darzustellen. Dieser letztgenannte Weg führt auf die sogenannte "transformationelle Semantik" von Programmiersprachen /21/.

## 5. Ein Beispiel

Mit dem folgenden Beispiel sollen die wesentlichen Elemente einer Breitbandsprache vorgestellt und in ihrem Gebrauch demonstriert werden. Wir stützen uns dabei auf die Sprache ABSTRACT-PASCAL/13,14/. ABSTRACT-PASCAL entstand ursprünglich als Erweiterung der Sprache PASCAL für Zwecke der automatischen Programmverifikation. Sie umfaßte anfangs daher nur Ausdrucksmittel für die systematische Programmannotation /1,2/. Genaueres Studium der Entwurfsprozesse /18/ führte zu der Erkenntnis, daß die Ebene der applikativen (funktionalen) Programmierung als Bindeglied zwischen Spezifikation und "reinen" PASCAL-Programmen (als Vertretern für prozedurale Programmierweise) einfach notwendig ist. Bestärkt durch entsprechende internationale Entwicklungen, wurde ABSTRACT-PASCAL schrittweise von einer "Spezifikations- und Programmiersprache" /13/ zu einer Breitbandsprache aufgerüstet.

In diesem Abschnitt soll ein Programm zur Berechnung der ganzzahligen Wurzel einer nichtnegativen ganzen Zahl entwickelt werden. Wir folgen dabei dem Ansatz von Dijkstra /9/ und Gries /12/. Am Ende des Abschnitts werden die Ausdrucksmittel der Sprache ABSTRACT-PASCAL in einer tabellarischen Übersicht zusammengefaßt.

### 5.1. Spezifikation der Aufgabenstellung

Am Anfang steht die formale Spezifikation der Aufgabe:

```
( 1 ) function isqrt(n: integer/ n≥0): integer
( 2 )   abstract isqrt(n)^2≤n<(isqrt(n)+1)^2 end
```

Bemerkungen:

1. Wir haben es hier mit einer deskriptiven Funktionsvereinbarung zu tun: die Funktion wird durch eine Aussage über die Funktion als Ganzes definiert. Allgemein sind deskriptive Funktionsvereinbarungen von der Gestalt

$$\underline{\text{function}}\ f(x{:}\ t/\ h_x){:}\ tt\ \underline{\text{abstract}}\ H_f\ \underline{\text{end}}\ ,$$

wobei $h_x$ den Definitionsbereich der Funktion beschreibt, und $H_f$ eine Aussage über die Funktion $f$ darstellt.

Deskriptive Funktionsvereinbarungen sind notwendig um Aussagen über Funktionen zu formulieren, die mehr als einen Funktionswert betreffen. g sei z.B. eine beliebige auf dem Intervall 0..100 monotone Funktion. Wir definieren dann:

```
predicate monoton( function f(x: integer): integer;
                                a, b: integer)
   = !all(m, n: integer/ a≤m≤n≤b): f(m)≤f(n) ;
function g(i: integer): integer
   abstract monoton(g, 0, 100) end
```

2. Die formalen Parameter der Funktion gelten innerhalb der Funktionsvereinbarung als generalisierte Variablen, d.h. Zeile (2) ist eine Kurzform für

$$\text{!all(n: integer/ } n \geq 0\text{): isqrt}(n)^2 \leq n < (\text{isqrt}(n)+1)^2$$

(lies: "für alle 'n' vom Typ 'integer' mit '$n \geq 0$' gilt ...").

3. Aus Gründen der leichteren Lesbarkeit wird in diesem Abschnitt durchgehend von der Exponentenschreibweise gebrauch gemacht. Korrekt müßte es anstelle von $x^2$ stets sqr(x) heißen. Ebenso werden wir für die ganzzahlige Division den Schrägstrich a/b anstelle von a div b verwenden. ■

Am Anfang des Programmentwurfs könnte ebenso eine applikative Funktionsvereinbarung stehen:

```
( 1 ) function isqrt(n: integer/ n≥0): integer
( 2') = the(i: integer/ i²≤n<(i+1)²)
```

Bemerkungen:

1. Applikative Funktionsvereinbarungen haben die Form

$$\text{function f(x: t/ } h_x\text{): tt} = e_x \quad ,$$

wobei $h_x$ wiederum den Definitionsbereich der Funktion beschreibt, und $e_x$ ein Ausdruck vom Typ tt mit der freien Variable x (vom Typ t) ist, der für alle x mit $h_x$ definiert ist. Applikative Funktionsvereinbarungen entsprechen der Wertverlaufsdefinition für Funktionen im mathematischen Sinne.

2. Funktionen müssen nicht notwendig determiniert sein. Der definierende Ausdruck $e_x$ kann sowohl deskriptive als auch applikative Ausdrucksmittel enthalten. In Zeile (2') haben

wir es mit einem solchen deskriptiven (abstrakten) Ausdruck zu tun. Man unterscheidet gewöhnlich drei Formen von abstrakten Ausdrücken:

| | |
|---|---|
| all(x: t/ $h_x$) | lies: "die Menge aller 'x' vom Typ 't' mit der Eigenschaft '$h_x$'" , |
| one(x: t/ $h_x$) | lies: "ein (beliebiges) 'x' vom Typ 't' mit '$h_x$'" und |
| the(x: t/ $h_x$) | lies: "dasjenige 'x' vom Typ 't' mit der Eigenschaft '$h_x$'" |

Diese abstrakten Ausdrücke entsprechen dem Mengenkonstruktor "{x: $h_x$}", dem Auswahloperator "$\mu$" bzw. dem Kennzeichnungsoperator "$\iota$" im geläufigen mathematischen Sinne.

3. Ein wichtiges Ausdrucksmittel zur Formulierung von Aussagen über Programmelemente sind Quantoren. Den Allquantor (!all) haben wir bereits benutzt. Man unterscheidet wiederum drei Arten von Quantoren:

| | |
|---|---|
| !all(x: t/ $h_x$): $hh_x$ | lies: "für alle 'x' vom Typ 't' mit '$h_x$' gilt die Bedingung '$hh_x$'" , |
| !one(x: t/ $h_x$) | lies: "es gibt ein 'x' vom Typ 't' mit der Eigenschaft '$h_x$'" und |
| !the(x: t/ $h_x$) | lies: "es gibt genau ein 'x' vom Typ 't' mit der Eigenschaft '$h_x$'" . |

Auch wenn es theoretisch möglich wäre, den Allquantor als alleinigen Quantor zuzulassen, und man sogar diesen auf Grund des Parameterkonzepts eliminieren könnte, hat sich diese Dreieinigkeit speziell in der Programmierung bewährt. Alles andere bedeutet nichts weiter als erhöhten Schreibaufwand und so etwas wie "logischen Harakiri". Die Analogie der Quantoren zu den abstrakten Ausdrücken und zu den Parameterabschnitten der Funktions- und Prozedurvereinbarungen ist natürlich beabsichtigt. ■

Mit der zweiten Fassung der Spezifikation haben wir schon einen gewaltigen Fortschritt gemacht: Wir wissen nun, daß es genau einen Wert mit der geforderten Eigenschaft gibt, und kennen eine obere und eine untere Schranke für den Funktionswert.

## 5.2. Der Algorithmus auf applikativem Niveau

Aus den bisher vorliegenden Informationen können wir die erste Fassung des Algorithmus herleiten. Die Grundidee ist simpel. In Anfängervorlesungen zur Analysis wird sie oft als Löwenfangmethode bezeichnet: das Intervall zwischen oberer und unterer Schranke wird solange halbiert bis man den gesuchten Funktionswert hinreichend genau bestimmt hat. Da wir nun mit ganzen Zahlen rechnen, erhalten wir als Genauigkeitsschranke, die den Abbruch des Verfahrens garantiert 1. Dafür müssen wir aber garantieren, daß ein Halbieren der Intervalllänge stets möglich ist. Am einfachsten wählt man hierzu die Anfangswerte für die obere und untere Intervallgrenze so, daß die Intervallänge gerade eine Zweierpotenz ist. Dieses Verfahren der Intervallschachtelung ist so allgemein, daß es sich lohnen würde, die entsprechenden Funktionen in einer Bibliothek von Standardmethoden bereitzustellen:

```
( 1) predicate power_of_2(x: integer)
( 2)   = !one(i: integer/ 2**i = x ) ;
( 3) function inversion( function f(x: integer): integer;
( 4)                       lwb, upb: integer
( 5)                     / monoton(f, lwb, upb) & f(lwb)≤0 &
( 6)                       f(upb)>0 & power_of_2(upb-lwb)
( 7)                     ): integer
( 8)   = if upb-lwb=1      {Intervallänge gleich 1}
( 9)     then lwb          {untere Grenze }
(10)     else if f(xxx)≤0
(11)          then inversion(f, xxx, upb)
(12)          else inversion(f, lwb, xxx)
(13)          fi
(14)     fi
(15)   where
(16)     function xxx: integer =(lwb+upb)/2
(17)   end {inversion}
```

Bemerkungen:

1. Die Funktion inversion berechnet die "ganzzahlige Nullstelle" der monoton wachsenden Funktion f im Intervall lwb..upb .

2. Die Funktion inversion ist durch einen bedingten Ausdruck definiert. Man unterscheidet zwei Formen bedingter Ausdrücke: den deterministischen bedingten Ausdruck (in Anlehnung an Algol):

if b then $e_1$ else $e_2$ fi

und den nichtdeterministischen bedingten Ausdruck (vgl. /9/):

if $b_1 \rightarrow e_1$ # ... # $b_n \rightarrow e_n$ fi

wobei die $b_i$ jeweils Boolesche Ausdrücke und die $e_i$ Ausdrücke von einem gemeinsamen Typ t , dem Resultattyp des bedingten Ausdrucks sein müssen. Bedingte Ausdrücke sind ein unentbehrliches Hilfsmittel zur Definition rekursiver Funktionen.
3. Die where-Klausel am Ende einer Funktions- oder Prozedurvereinbarung umfaßt Vereinbarungen, die lokal zu der jeweiligen Funktion bzw. Prozedur sein sollen. Sie stellt gewissermaßen einen "Nachtrag" zum Deklarationsteil einer Vereinbarung dar. In vielen Fällen erhöht sich durch ein solches "Nachstellen" einiger lokaler Objekte die Lesbarkeit, indem nämlich die wesentliche Teile der Vereinbarung (Kopf und Körper) nicht durch unwesentliche Festlegungen getrennt werden.
4. Prädikate werden in ABSTRACT-PASCAL wie Boolesche Funktionen behandelt, mit dem Unterschied, daß Prädikate in der Regel (d.h. außer zu Testzwecken) nicht implementiert werden. Genau wie für Funktionen gibt es deskriptive und applikative Prädikat vereinbarungen. ☒

Unter Benutzung der Funktion inversion erhalten wir eine erste algorithmische Darstellung für isqrt

```
( 1) function isqrt(n: integer/ n≥0): integer
( 2)   = inversion(g, 0, obere_schranke)
( 3)   where
( 4)     function g(x: integer): integer = x^2-n ;
( 5)     function obere_schranke: integer
( 6)       = 2**one(i: integer/ (2**i)^2≥n)
( 7)   end {isqrt}
```

Durch Teilberechnung (Einsetzen und Auflösen von g und xxx ) und anschließendes Zusammenfassen zu einer neuen (Hilfs-)funktion invert_sqr entsteht daraus

```
( 1) function isqrt(n: integer/ n≥0): integer
( 2)   = invert_sqr(0, obere_schranke)
( 3)   where
( 4)   function invert_sqr( lwb, upb: integer
( 5)                      / lwb^2-n ≤0 & upb^2-n >0 &
( 6)                        power_of_2(upb-lwb)
( 7)                      ): integer
( 8)     = if upb-lwb=1
( 9)       then lwb
(10)       else if ((lwb+upb)/2)^2-n ≤0
(11)            then invert_sqr( (lwb+upb)/2, upb)
(12)            else invert_sqr( lwb, (lwb+upb)/2)
(13)            fi
(14)       fi;
(15)   function obere_schranke: integer
(16)     = 2**one(i: integer/ 4**i >n)
(17)   end {isqrt}
```

Durch Anwendung des allgemein gültigen Transformationsschemas WIp (while-introduction for partially defined functions) entsteht die erste prozedurale Version der Funktion isqrt (vgl. Abb.7)

```
function F(x: T/ D(x)): TT
  = if B(x)
    then H(x)
    else F(G(x))
    fi
```

```
function F(x: T/ D(x)): TT
  var xx: T;
  begin xx:=x;
        while not B(xx) do
          begin assert(D(xx));
                xx:=G(xx)
          end;
        F:=H(xx)
  end
```

```
function D(x: T):Boolean;
function B(x: T/D(x)):
                    Boolean;
function G(x: T/D(x)): T;
function H(x: T/D(x)): TT;
```

Abb.7 Transformationsschema WIp

5.3. Der Algorithmus auf prozeduralem Niveau

```
( 1) function isqrt(n: integer/ n≥0): integer;
( 2)   var lwb, upb: integer;
( 3)   begin (lwb,upb) := (0, obere_schranke);
( 4)         while upb-lwb≠1 do
( 5)           begin assert( lwb^2≤n<upb^2 & power_of_2(upb-lwb) );
( 6)             if ((lwb+upb)/2)^2≤n
( 7)             then (lwb,upb) := ((lwb+upb)/2, upb)
( 8)             else (lwb,upb) := (lwb, (lwb+upb)/2)
( 9)           end;
(10)         isqrt:=lwb
(11)   end
(12)   where
(13)     function obere_schranke: integer
(14)       = 2**one(i: integer/ 4**i>n)
(15)   end {isqrt}
```

Bemerkungen:

1. Das auffälligste an dieser Implementation dürfte die systematische Verwendung simultaner Wertzuweisungen sein. Besonders Dijkstra /9/ und Gries /12/ haben auf die methodologischen Vorteile dieser Anweisungstypen hingewiesen. In unserem Beispiel sind sie dadurch entstanden, daß die Transformationsregel nicht auf eine Funktion mit einem Parameter sondern auf eine Funktion mit mehreren (2) Parametern angewendet wurde. Das Ergebnis der Programmtransformation wird noch deutlicher, wenn man sich die bedingte Anweisung der Zeilen (6-8) als eine Zuweisung eines bedingten Ausdrucks vorstellt:

```
(lwb,upb) := if ((lwb+upb)/2)^2 ≤n
             then ((lwb+upb)/2; upb)
             else (lwb, (lwb+upb)/2)
             fi
```

2. Auch auf prozeduralem Niveau des Gesamtalgorithmus können einzelne Komponenten applikativen bzw. deskriptiven Charakter tragen. In unserem Fall der Teilausdruck obere_schranke . ■

Der Algorithmus ist jedoch noch nicht genügend effektiv: in jedem Zyklusdurchlauf muß das Quadrat einer Zahl neu berechnet werden. Durch formale Integration des Zykluskörpers erhält man die Variablensubstitution

$$(p,q,r) =_{def} (lwb^2,\ lwb*(upb-lwb),\ (upb-lwb)^2)$$

Wir wenden diese Variablensubstitution auf das Programm an und erhalten:

```
( 1) function isqrt(n: integer/ n≥0): integer;
( 2)   var p,q,r: integer;
( 3)   begin (p,q,r) := (0, 0, obere_schranke²);
( 4)         while r≠1 do begin
( 5)           assert( p≤n<p+2*q+r & q²=p*r & power_of_4(r) );
( 6)             if p+q+r/4≤n
( 7)             then (p,q,r):= (p+q+r/4, q/2+r/4, r/4)
( 8)             else (p,q,r):= (p, q/2, r/4)
( 9)           end;
(10)         isqrt:=q
(11)   end
(12)   where
(13)     function obere_schranke: integer
(14)       = 2**one(i: integer/ 4**i>n);
(15)     predicate power_of_4(x: integer)
(16)       = !one(i: integer/ x=4**i )
(17)   end {isqrt}
```

Als nächstes betrachten wir nun die bedingte Anweisung im Zykluskörper. Es scheint offensichtlich, daß sich durch Sequentialisierung und Ausklammerung (Vorausberechnung) von Teilausdrücker einiges vereinfachen läßt:

```
var s: integer;
begin r:=r/4; s:=p+q+r; q:=q/2;
      if s≤n
      then begin p:=s; q:=q+r end
end
```

Als letzten Schritt müssen wir jetzt noch den Teilausdruck obere_schranke implementieren. Wir geben gleich die iterative

Fassung an:

```
( 1) function isqrt(n: integer/ n≥0): integer;
( 2)   var p,q,r,s: integer;
( 3)   begin p:=0; q:=0; r:=1;
( 4)         while r≤n do
( 5)           begin assert( power_of_4(r) ); r:=r*4 end
( 6)         while r≠1 do
( 7)           begin assert( p≤n<p+2*q+r  & q^2=p*r &
( 8)                         power_of_4(r) );
( 9)                 r:=r/4; s:=p-q+r; q:=q/2;
(10)                 if s≤n
(11)                 then begin p:=s; q:=q+r end
(12)           end;
(13)         isqrt:=q
(14)   end
(15)   where
(16)     predicate power_of_4(x: integer)
(17)         = !one( i: integer/ x=4**i )
(18)   end  {isqrt}
```

Dieses Programm zur Berechnung der ganzzahligen Wurzel erfüllt nun wirklich alle Anforderungen, die wir an ein effektives und zugleich wohl strukturiertes Programm stellen. Doch es hat es in sich. Vor einigen Jahren stellte uns A.Blikle die Aufgabe, dieses Programm (ohne die Zyklusinvarianten versteht sich) mit Hilfe von Verifikationsstrategien "zuknacken" - vergeblich waren alle Bemühungen. Probieren Sie es aus: stellen Sie einem Ihrer Kollegen die Knobelaufgabe, die Funktionsweise dieses Programms zu ergründen!

Überschaut man rückblickend den Entwurfsprozess, so sind nur an zwei Stellen wirklich wichtige Entscheidungen getroffen worden: zuerst bei der Auswahl der Methode der Intervallschachtelungen, später dann der Übergang zum Variablensystem (p,q,r). Derartige Entscheidungen fallen nicht vom Himmel, sondern erfordern einen gehörigen Schuß Intuition und auch Mut, einen Ansatz einfach auszuprobieren. Beides kann uns der Rechner nicht abnehmen. Programmieren ist und bleibt immer eine krea-

| | deskriptives Sprachniveau | applikatives Sprachniveau | imperatives Sprachniveau |
|---|---|---|---|
| *Typen* | algebraische Spezifikation | Typrestriktionen<br>- logisch (einschränkende Bedingungen)<br>- algebraisch (Generierungsprinzip) | erweitertes Sortiment an Standard- und strukturierten Typen |
| *Prädikate* | Quantoren | bedingte Ausdrücke | sequentielle Boolesche Operationen |
| | deskriptive Prädikatvereinbarung<br>**predicate** p(x:t)<br>**abstract** $H_p$ **end** | applikative Prädikatvereinbarung<br>**predicate** p(x:t)<br>= $h_x$ | imperative Prädikatvereinbarung<br>**predicate** p(x:t)<br>**begin** ... p:=..<br>**end** |
| *Funktionen* | abstrakte Ausdrücke | bedingte Ausdrücke | erweitertes Sortiment an Standardfunktionen und -operatoren |
| | deskriptive Funktionsvereinbarung<br>**function** f(x:t)<br>**abstract** $H_f$ **end** | applikative Funktionsvereinbarung<br>**function** f(x:t)<br>= $e_x$ | imperative Funktionsvereinbarung<br>**function** f(x:t)<br>**begin** ... f:=..<br>**end** |
| *Prozeduren* | Programmannotationen | simultane Wertzuweisung | verallgemeinerte Ablaufstrukturen |
| | deskriptive Prozedurvereinbarung<br>**procedure** p(x:t)<br>**entry** $h_x$<br>**exit** $hh_x$ | applikative Prozedurvereinbarung<br>**procedure** p(x:t)<br>x ← f(x,y) | imperative Prozedurvereinbarung<br>**procedure** p(x:t)<br>**begin** ... x:=..<br>**end** |

Abb. 8 Ausdrucksmittel von Breitbandsprachen

tive Tätigkeit - es kommt nur darauf an, die "nervtötenden" monotonen Teilprozesse zu bezwingen.

Breitbandsprachen und Programmtransformationen sind vielleicht ein Weg, um auf dieser Strecke weiter zu kommen. Den unbestreitbaren Vorteilen, die sich aus dem Einsatz von Breitbandsprachen für die gesamte Technologie der Programmentwicklung ergeben, stehen jedoch erhebliche Akzeptanzprobleme gegenüber. Sie resultieren vor allem daraus, daß derartige neue, formale Sprachen als Kommunikationsmittel weniger geeignet sind als die althergebrachten, meist informalen (grafischen oder auch tabellarischen), für konkrete Anwendungsfälle bereits aufbereiteten sprachlichen Mittel. Eine weitere Ursache dürfte jedoch auch darin liegen, daß die zugrunde liegenden Kalküle der formalen Logik und der diskreten Mathematik von den heute in der Praxis tätigen Programmierern nicht (mehr) im notwendigen Maße beherrscht werden.

Breitbandsprachen werden sich nur dann einen Platz als Programmierhilfsmittel erobern und behaupten können, wenn ihre Anwendung methodisch in geeigneter Weise aufbereitet wird, und wenn die bislang nur für kleine Beispiele demonstrierte Rechnerunterstützung auf der Basis von Programmtransformationen /11/ ökonomisch für breite Anwendungen vertretbar wird.

## Literaturhinweise


/ 1/ Bär,D.: Beiträge zur strukturierten Programmierung und zur Programmverifikation. AdW, Zentrum f.Rechentechnik 1978

/ 2/ Bär,D.: Abstractions in PASCAL programming. Humboldt-U., Sektion Mathematik, Seminarbericht Nr.13, 1979

/ 3/ Bauer,F.L.: Program Development by Stepwise Transformation - The Project CIP. Lect.Notes Comp.Sci.69(1979)237-266

/ 4/ Bauer,F.L.: Systematics of Transformation Rules. Lect.Notes Comp.Sci.69(1979)273-289

/ 5/ Bauer,F.L. u.a.: Towards a Wide-Spectrum Language to Support Program Specification and Program Development. Lect.Notes Comp.Sci.69(1979)543-552

/ 6/ Bauer,F.L.; Wössner,H.: Algorithmische Sprache und Programmentwicklung. Springer-Verlag, Berlin u.a. 1981

/ 7/ Bergstra,J.A.; Terlouw,J.: A Characterization of Program Equivalence in Terms of Hoare's Logic. Manuskript.

/ 8/ Burstall,R.M.; Darlington,J.: A Transformation System for Developing Recursive Programs. JACM 24(1977)1,44-67

/ 9/ Dijkstra,E.W.: A Discipline of Programming. Prentice-Hall, Englewood Cliffs 1976

/10/ Dömölki,B.: An Example of Hierarchical Program Specification. Lect.Notes Comp.Sci.86(1980),333-353

/11/ Feather,M.S.: A System for Assisting Program Transformation. ACM Trans.Prog.Lang.and Systems 4(1982)1,1-20

/12/ Gries,D.: The Science of Programming. Springer-Verlag, Berlin u.a. 1981

/13/ Horn,Ch.: Programmentwurf und -spezifikation in ABSTRACT-PASCAL. Wiss.Z.Humboldt-U., Math.-Nat.R. 30(1981)5,423-430

/14/ Horn,Ch.: Die Spezifikationssprache ABSTRACT-PASCAL. Rechentechnik/Datenverarbeitung (Beiheft) 18(1981)3,22-27

/15/ Horn,Ch.: Einsatz rechnergestützter Methoden in der Softwareproduktion. edv-aspekte 1(1982)3,9-12

/16/ Howden,W.E.: Contemporary Software Development Environments. CACM 25(1982)5,318-329

/17/ Hünke,H. (ed.): Software Engineering Environments. North-Holland, Amsterdam 1981

/18/ Klehn,N.: Die Anwendung von ABSTRACT-PASCAL beim Programmentwurf. Diplomarbeit. Humboldt-U., Sektion Mathematik 1981

/19/ Loveman,D.B.: Program Improvement by Source-to-Source Transformations. JACM 24(1977)1,121-145

/20/ Moriconi,M.: A Designers/Verifyers Assistant. IEEE Trans.Software Engineering 5(1979)4,387-401

/21/ Pepper,P.: A Study on Transformational Semantics. Lect.Notes Comp.Sci.69(1979),322-405

/22/ Vogel,J.: Datenverarbeitungsprojektierung - Stand und Entwicklung. edv-aspekte 1(1982)3,2-9

/23/ Wegbreit,B.: Goal-directed Program Transformations. IEEE Trans.Software Engineering 2(1976)2,69-80

/24/ Wilander,J.: An interactive programming system for PASCAL. BIT 20(1980)2,163-174

/25/ Wössner,H.: Special Transformation Techniques. Lecture Notes Comp.Sci. 69(1979),290-321

Verfasser:

Dipl.-Math.Christian Horn

1035 Berlin

Dolziger Str.27

Vack, Gert-Ulrich

# Verwendung des SM4-Makroassemblers als Übersetzer für einfache problemorientierte Sprachen

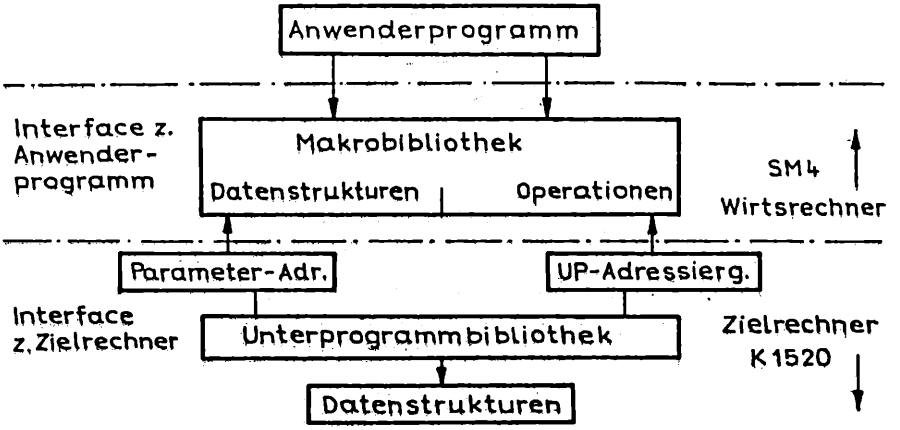
## 1. Aufgabenstellung

In der DDR werden in breitem Maße Mikrorechner mit einer Verarbeitungsbreite von 8 Bit auf Basis des CPU-Schaltkreises U 880 eingesetzt. Ein wichtiger Aspekt ist dabei die Erstellung der Anwendungssoftware.

Zur Mikrorechnerprogrammierung können Entwicklungssysteme oder Wirtsrechner eingesetzt werden. In den Systemunterlagen werden vor allem Assembler, z.T. mit Makroprozessor angeboten. Compiler für höhere Programmiersprachen stehen kaum zur Verfügung bzw. sind nur eingeschränkt nutzbar: BASIC kann für Echtzeitaufgaben oftmals nicht eingesetzt werden [1]; ein PASCAL-Compiler wird derzeitig erarbeitet [2]; PL-M ist für den 8080 konzipiert [3] (beim U880 werden bei der Nutzung von PL-M Leistungspotenzen verschenkt). Darüber hinaus gibt es Arbeiten zu CDL-2 [4] und PL-Z [5]. Aus verschiedenen Gründen werden diese Sprachen für die Programmierung praktisch wenig genutzt (Verfügbarkeit, Vertrieb, Eigenschaften des Sprachkonzeptes, Eigenschaften der Compiler). Demgegenüber steht für viele Anwendungsfälle die Forderung nach problemorientierte Sprachen. Die Implementierung von Übersetzern für problemorientierte Sprachen auf Basis der Assemblersprache erfordert einen hohen Aufwand. Im folgenden soll eine andere, weniger aufwendige Implementierungsmöglichkeit vorgestellt werden, bei der die Makrotechnik eines entsprechenden leistungsfähigen Makroassemblers ausgenutzt wird. Ein solcher Makroassembler ist innerhalb des SKR auf den Rechnern SM 3/SM 4 bzw. K 1600 gegeben [6].

## 2. Lösungsprinzip

Bei der Implementierung eines Übersetzers nach dem vorgeschlagenen Prinzip ergibt sich folgender Aufbau der Programmebenen:




In der untersten Ebene werden die Datenstrukturen definiert, mit denen eine programmtechnische Lösung der Problemklasse realisiert werden kann. Für regelungstechnische Aufgabenstellungen müssen z.B. die Zahlenformate der Arithmetik und die Darstellung der Zahlen als Bitmuster festgelegt werden.

Auf diesen Datenstrukturen werden Funktionen definiert (z.B. arithmetische Operationen). Diese Funktionen müssen als Programme für den Zielrechner implementiert werden (Unterprogrammbibliothek). Auf die Datenstrukturen kann dabei nur über die entsprechenden Funktions-Unterprogramme zugegriffen werden (objektorientierter Entwurf).

Der Übersetzer muß in der Lage sein, den im problemorientierten Anwenderprogramm formulierten Algorithmus in eine Sequenz ausführbarer Befehle zu übersetzen, die die Parameterübergaben zwischen den Unterprogrammen sowie den Aufruf der Unterprogramme realisieren. Dazu genügen einfache Befehle (Unterprogramm-Aufruf, Ladebefehle mit indirekter oder indexierter Adressierung usw.), die sich für andere Zielrechnertypen problemlos austauschen lassen.

Auf dem Wirtsrechner werden alle innerhalb der Sprache definierten Anweisungen in Form von Makros vereinbart, in denen jeweils die Befehlscodes der Maschinenbefehle des Zielrechners als Byte- bzw. Adreßkonstanten fixiert sind.

Das Anwendungsprogramm wird als Folge von Makrorufen geschrieben. Bei der Übersetzung mit dem Makroassembler erfolgt die Syntaxprüfung mit den im Assembler implementierten Mitteln. Darüber hinaus können syntaktische Fehler nur erkannt werden, wenn bei der Implementierung der Makrobibliothek entsprechende Kontrollen vorgesehen wurden.

Zusammenfassen seien die verschiedenen Entwurfs- und Programmierschritte genannt:

1. Definition der Datenstrukturen im Zielrechner
2. Definitionen der Operationen im Zielrechner, die auf die Datenstrukturen gerichtet sind
3. Definition des Interface zum Wirtsrechner
4. Implementierung der Programme zu 1. bis 3.
5. Implementierung eines Laders für den Zielrechner bzw. eines Nachübersetzers für den Wirtsrechner (zum Laden des Task-Image-Files im Mikrorechner bzw. zur Generierung des Standard-Lade-Formates für Mikrorechner im Wirtsrechner).

Machen sich im Prozeß der Anwenderprogrammierung Änderungen / Erweiterungen eines einmal implementierten Konzeptes erforderlich, so müssen in der Regel nur die entsprechenden neuen Datenstrukturen und Operationen bereitgestellt werden (Ergänzungen der Unterprogrammbibliothek und Erweiterungen der Makrobibliothek). Damit ist prinzipiell eine Kompatibilität der Programme an der Schnittstelle zum Anwender gesichert.

## 3. Ein Beispiel: Nutzung einer Gleitkommaarithmetik

Für Echtzeit-Steuerungsaufgaben wird eine schnelle, registerorientierte Gleitkommaarithmetik benötigt [7]. Als Datenformat wird eine 3-Byte-Darstellung (2 Byte Mantisse, 1 Byte Exponent) festgelegt.
Zweistellige Gleitkommaoperationen (z.B. Addition) haben zwei Eingabeoperanden und einen Ausgabeoperanden. Operanden werden nicht direkt übergeben, sondern es werden die Adressen der Operanden übergeben. In einem Assemblerprogramm für den K 1520 könnte folgende Befehlsfolge vereinbart werden:

```
; VEREINBARUNG DER DATENSTRUKTUR F. 1. OPER.
AX1:    DA      X1      ; ADRESSE DES 1. OPER.
X1:     BER     3       ; 3 BYTE FUER 1. OPER.
; VEREINBARUNG DER DATENSTRUKTUR F. 2. OPER.
AX2:    DA      X2
X2:     BER     3
; VEREINBARUNG DER DATENSTRUKTUR F. ERGEBNIS
AERG:   DA      ERG
ERG:    BER     3
; BERECHNUNG VON ERG := 1. OPER. + 2. OPER.
; PARAMETERÜBERGABEN
        LD      BC,(AX1)
        LD      DE,(AX2)
        LD      HL,(AERG)
; AUFRUF DES UNTERPROGR. ZUR GLEITKOMMAADD.
        CALL    FTADD
```

Mit zwei Makros auf Ebene der SM4-Assemblersprache lassen sich die Sequenzen zur Vereinbarung von REAL-Variablen und zum Aufruf des Unterprogrammes erzeugen:

```
        .MACR   REAL NAME
NAME:   .WORD   .+2     ; ADRESSZEIGER ZUR VARIABLEN
        .BLKB   3       ; 3 BYTE FUER VARIABLE
        .EVEN
        .ENDM

        .MACR   ADD     OP1,OP2,Z
        .EVEN
        .WORD   45755,OP1       ; GENERIER. V. LD BC,(OP1)
        .WORD   55755,OP2       ; GENERIER. V. LD DE,(OP2)
        .WORD   25000,Z         ; GEN. V. NOP
                                ;         LD HL,(ERG)
        .WORD   146400,FTADD    ; GEN. V. NOP
                                          CALL FTADD
```

(Die Einfügung von .EVEN sowie von Leerbefehlen NOP ist erforderlich, weil in der Assemblersprache der SM4 Adresswerte nur auf geraden Speicheradressen vereinbart werden können. Bei Verwendung eines weiteren Makros, der die Vereinbarung von Worten auch

auf ungeraden Adressen realisiert, kann die Einfügung entfallen.)

Das Anwendungsprogramm kann unter Verwendung dieser Makros wie folgt geschrieben werden:

```
        .MCALL  REAL,ADD
        REAL    X1
        REAL    X2
        REAL    ERG


        .
        ADD     X1,X2, ERG


        .
        .END
```

In gleicher Weise können Makros für andere Datentypen und Operationen vereinbart werden.

4. Implementierungsvarianten

Das dargestellte Prinzip kann mit verschiedenen Implementierungsvarianten realisiert werden. Die im Beispiel verwendete indirekte Operandenadressierung ist für die Vereinbarung dynamischer Felder erforderlich. Außerdem ist die indirekte Adressierung Voraussetzung für die Überprüfung von Zugriffsrechten in den Unterprogrammen.

Wenn Anwenderprogramme ausschließlich aus Unterprogrammaufrufen bestehen, kann der Befehlscode für den Befehl 'CALL' eingespart werden. Auch die sich wiederholenden Befehlscodes für die Indirekt-Ladebefehle können entfallen. Das Anwenderprogramm besteht dann nur noch aus einer Folge von Adressen, die zu den Parametern und den Unterprogrammen vermitteln. Die Programmsteuerung muß von einem Interpreter übernommen werden. Für den Mikrorechner K 1520 benötigt ein derartiger Interpreter pro Schleifendurchlauf (d.h. für eine zu interpretierende Adresse) ca. 60 Takte gegenüber 17 Takte bei einer direkten Programmierung über CALL-Befehle. Die Speicherplatzeinsparung beträgt 30...50% gegenüber einer in Makros formulierten vollständigen Assemblerprogrammierung. Dieses

Implementierungsprinzip wird mit "threaded code" bezeichnet und im Zusammenhang mit leistungsfähigen, maschinenunabhängigen und wenig speicherplatzintensiven Compilerlösungen diskutiert [8].

Wenn die Anzahl der Operanden und Funktionen auf weniger als 256 beschränkt ist, kann ein weiteres Niveau der indirekten Adressierung zwischengeschaltet werden. Parameter und Operationen werden dabei nur noch durch ein Byte charakterisiert. Der Interpretationsaufwand liegt bei etwa 120 Takten je Schleife. Der Speicherplatzbedarf des Anwendungsprogrammes reduziert sich um ca. 50 bis 70%. Wenn die Unterprogramme des Basis-Software-Paketes (unterste Softwareebene in Bild 1) verschieblich programmiert sind, kann die Makrobibliothek um die Maschinenbefehle (Codes) der Unterprogramme erweitert werden. Bei jeder erstmaligen Verwendung der Makros wird das erforderliche Unterprogramm in einem Bibliotheksbereich während der Assemblierung generiert. Damit erhält man eine abgeschlossene, redundanzfreie Programmversion für das Mikrorechner-Anwendungssystem.

## 5. Vor- und Nachteile der vorgeschlagenen Lösung

Die vorgeschlagene Lösung hat folgende Vorteile:

1. Ähnlich wie in einigen modernen Programmiersprachen (CDL-2)[4] wird der Programmierer zu einer logischen Vorgehensweise bei der Problemspezifikation und bei der Programmierung gezwungen, wodurch saubere Schnittstellendefinitionen entstehen. Auf den verschiedenen Entwurfsebenen werden dabei Entwurfsentscheidungen gegenüber darüberliegenden Ebenen verborgen (objektorientierter Software-Entwurf). Damit wird insbesondere den Forderungen nach Modularität, Portabilität und Erweiterbarkeit von Programmen Rechnung getragen. Bei Übertragung von Anwenderprogrammen auf andere Zielrechner müssen die entsprechenden Unterprogramme für den Zielrechner erstellt sowie die Maschinencodes in den Operations-Makros geändert werden. Eine Änderung der Anwendersoftware ist nicht erforderlich.
2. Die Implementierung des Übersetzers wird vereinfacht. Der SM4-Makroassembler übernimmt die Syntaxprüfung.
3. Der Programmierer hat freie Hand bei der Implementierung von Compile-Time-(in den Makros realisiert) bzw. Run-Time-(in den

Unterprogrammen realisiert) Fehlerkontrollen. Das kann bis zur Überwachung von Zugriffsrechten auf bestimmte Datentypen seitens der Operationen gehen.

Folgende Nachteile sind festzustellen:

1. Die sprachlichen Ausdrucksmittel von auf diesem Wege implementierten Übersetzern für einfache problemorientierte Sprachen sind gering.
2. Bei der Implementierung beliebiger Datenstrukturen und Operationen stehen nur wenige Freiheitsgrade zur Verfügung.
3. Die Quelltextgestaltung ist durch die Anforderungen aus der Makro-Assemblersprache bereits fixiert (z.B. sind Namen nur auf max. 6 Zeichen auswertbar).

Trotz dieser Nachteile wird eingeschätzt, daß für einfache Probleme die vorgeschlagene Lösung vorteilhaft ist - insbesondere wegen der unkomplizierten und schnellen Implementierbarkeit der Übersetzer.

## 6. Literatur

[1] BASIC-Interpreter für den CAMAC-Mikrorechner AMCA-80. Institut für Elektronenphysik der Akademie der Wissenschaften der DDR

[2] Wiederänders: Seminarbericht PAMIR-PASCAL für Mikrorechner. Humboldt-Universität Berlin, Sektion Mathematik

[3] PL/M-80 Programming Manual. Intel Corporation 1978

[4] Schilling, A.: Erstellung von Kleinrechner-Software mittels CDL-2 auf ESER-Rechnern. Berlin Informatik-Tage 1982

[5] ZILOG-Betriebssystem RIO. Dokumentation

[6] Horn, T.; Kofer, C.: Handbuch der SKR-Programmierung Einführung in die Makroassemblersprache MACRO-SM und Anwendung des allgemeinen Plattenbetriebssystems DOS-SM für die Rechenanlagen SM 3 und SM 4. Heft 1 der Schriftenreihe "Informationsverarbeitung im Hoch- und Fachschulwesen"

[7] Chamrad, V.: Floating-Point Subroutines for 8080 Microprocessors (User's Guide). Czechoslovak Academy of Sciences. Report No. 873, Prague 1978

[8] Kogge, M.P.: An architectural trail to threaded code systems. COMPUTER (15) 1982, S. 22-32

Verfasser:
Dipl.-Ing. Gert-Ulrich Vack
Akademie der Wissenschaften der DDR - Zentralinstitut für Kybernetik und Informationsprozesse, Institutsteil Dresden
8027 Dresden, Haeckelstr. 20

Nikulski, Ernst

# TESYS - Ein technologisches System zur Software-Entwicklung

## 1. Einleitung

Der ständig steigende Bedarf an Software erfordert eine dementsprechende Produktivitätssteigerung durch Rationalisierung und Effektivierung auf dem Gebiet der Softwareentwicklung. Das wissenschaftliche Aufbereiten dieses Gebietes im Sinne einer Ingenieurwissenschaft führte zu einer Anzahl neuer Methoden, Verfahren und Werkzeugen. Ihre Anwendung sowie das Nutzen vorhandener Lösungen im Sinne standardisierter Bausteine führt jedoch nur dann zu den beabsichtigten Zielen, wenn die Mittel aufeinander abgestimmt sind, d. h. konfliktfrei miteinander korrespondieren und auch an die spezifischen Bedingungen des Anwenders anpaßbar sind. TESYS stellt eine solche rechnergestützte Lösung für die Zielsprachen PL1 und COBOL in Form von Prozessoren dar.

## 2. Bestandteile von TESYS

Eine Übersicht über die Bestandteile von TESYS gibt Bild 1. Eine wichtige programmtechnische Basis stellt der Makroprozessor dar. Er ermöglicht sowohl die angepaßte Nutzung der in der Methodenbank mitgelieferten Systembestandteile durch die übrigen Prozessoren, als auch das Speichern und Aufrufen von Anwenderalgorithmen. Sie können im Sinne bekannter Mittel der Makrotechnik modifiziert und auch entsprechend ihrer Zugehörigkeit zu TESYS-Bausteinen umgeordnet werden. Der Makroprozessor kann auch eigenständig für beliebige Zielsysteme, z. B. als Job-Control-Generator, genutzt werden.

Der Programmstrukturprozessor generiert auf der Grundlage der Zielsprachendefinition ein Programmskelett, in das an markierten Stellen Bausteine eingefügt werden. In die Bausteine zu übernehmende Quelltexte oder weitere Bausteine werden durch den Prozessor aus dem Eingabestrom gesammelt und mit Hilfe der Namenskonvention in die vorgesehene Stelle eingeordnet. Bausteinmarken können auch anwenderspezifisch gesetzt werden. Für Programme mit Phasenstruktur wird ein Phasenskelett bereitgestellt.

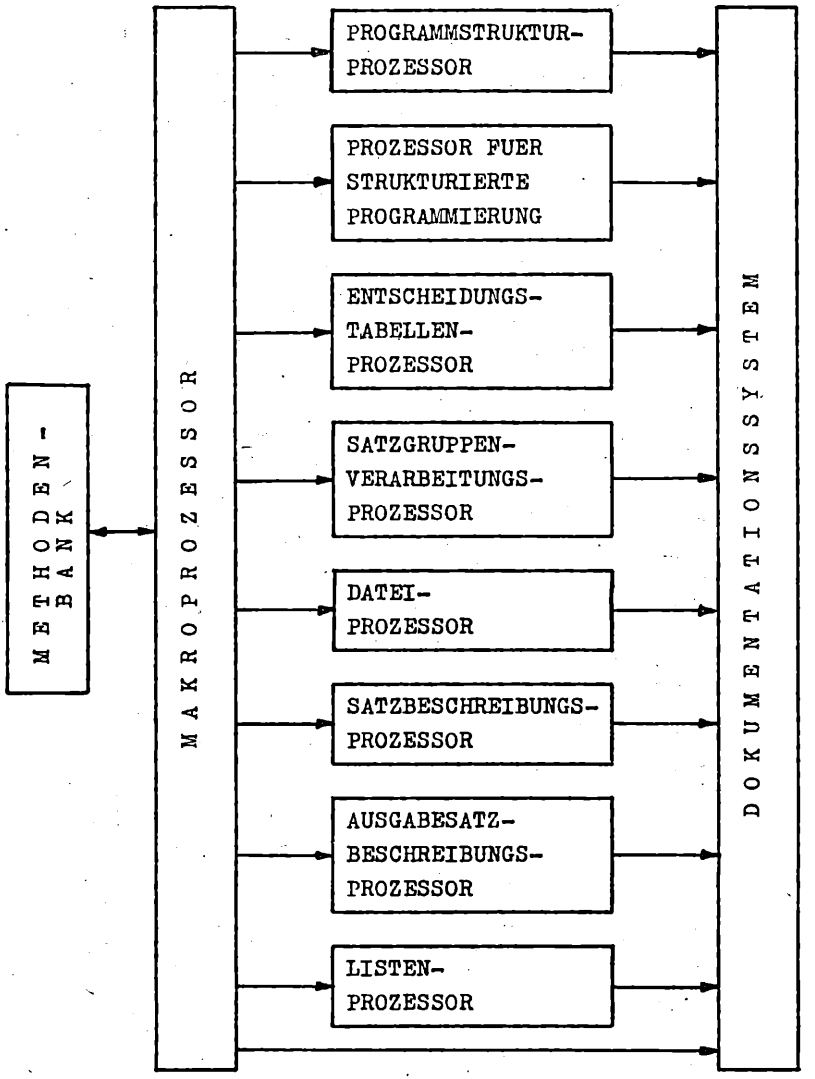



Bild 1 Die Bestandteile von TESYS

Beim Prozessor für Strukturierte Programmierung können mit einem Pseudocode die Konstrukte
- Reihung
- Fallunterscheidung
- Wiederholung und
- Verlassen

genutzt werden. Da auch freie Texte zugelassen sind, können strukturierter Entwurf und strukturierte Dokumentation auf logische Richtigkeit maschinell getestet werden. Für die Codegenerierung dürfen Strukturblöcke nur noch Pseudocode und Zielsprachencode enthalten.

Datenorientiertes Entwerfen und Implementieren nach Jackson einschließlich Programmierversion und Backtracking sind durch Nutzen von Programmstrukturprozessor und Prozessor für Strukturierte Programmierung möglich.

Der Entscheidungstabellenprozessor führt auch für freisprachlich formulierte Bedingungen und Aktionen die formale Strukturanalyse und Vollständigkeitsprüfung aus. Zur Generierung von Quellcode müssen Bedingungen und Aktionen in der Zielsprache oder als Pseudocode vorliegen. Entscheidungstabellen können sowohl eigenständig, als auch als Block der Strukturierten Programmierung eingesetzt werden.

Der Satzgruppenverarbeitungsprozessor erzeugt für die Verarbeitung von logisch sequentiellen Dateien auf der Grundlage des Wechsels im Gruppenbegriff Programmskelette und fügt die vom Anwender codierten Bausteine anhand vereinbarter Namen in das jeweils erzeugte Skelett ein. Das Mischen von Dateien und das Verarbeiten von Sekundärdateien bei Gruppenwechsel der steuernden Datei ist möglich.

Mit dem Dateiprozessor kann die Datenunabhängigkeit der Verarbeitung gesichert werden. Durch den Makroaufruf, der Definitionen und Programmskelette erzeugt, wird eine funktionsorientierte Schnittstelle zum Anwender gebildet. Werden die Namenskonventionen für Daten, Routinen und Bausteine eingehalten, sind beliebige Dateisysteme anschließbar.

Datensätze werden in einfacher Syntax beschrieben. Der Datensatzbeschreibungsprozessor erzeugt zielsprachengerechte Definitionen, maschinell auswertbare Dokumentationssätze und formatierte Listen.

Darüber hinaus erzeugt der Ausgabesatzbeschreibungsprozessor auch Übertragungsanweisungen, wenn die Herkunft der Elemente eines Ausgabesatzes als Eingabedatum, Rechenergebnis oder Konstante angegeben ist. Der Anwendung dieses Prozessors sind kaum Grenzen gesetzt, da Definitionen und Verarbeitungsanweisungen der Zielsprache eingesetzt werden können.

Der Listenprozessor ermöglicht das Beschreiben von Zeilen für Listenkopf, Listenfuß, Seitenkopf, Seitenfuß sowie von gruppenabhängigen Zeilen und Einzelzeilen. Spaltensummenbildung, Tabellenausgabe und gruppenabhängige Ausgabeunterdrückung ist möglich. Sehr günstig ist die Ausgabe von Pseudolisten anhand der Definition.

Mit den Prozessoren des Dokumentationssystems können hierarchisch strukturierte Dokumentationen, Verzeichnisse, Verwendungsnachweise u. a. ausgegeben werden. Die Dokumentationen können aus Beständen der Textbibliothek, Programmbibliothek und Dokumentationsbibliothek zusammengetragen werden. Sie werden mit Seitennummerierung, Kapitelüberschriften, Sachwortverzeichnis und Inhaltsverzeichnis versehen. Verwendungsnachweise u. ä. lassen sich aus den von den übrigen Prozessoren erzeugten Dokumentationssätzen ableiten. Die Ausgabe des Dokumentationssystems kann über ein Konvertierungprogramm dem Textverarbeitungssystem TVS/S zugeführt werden.

## 3. Wirkungsweise von TESYS

Die Prozessoren werden über TESYS-Anweisungen aktiviert. Sie stellen Anweisungen mit großer Mächtigkeit, d. h. hohem Problemgehalt dar. Parameter, die Standardwerten entsprechen, brauchen nicht spezifiziert zu werden. Es gibt System-, Datei- und Dokumentationsparameter, denen Standardwerte zugewiesen werden können. Diese lassen sich beim TESYS-Aufruf oder in den TESYS-Anweisungen modifizieren.

Die Prozessoren erzeugen entweder parametergesteuert direkt Code oder bilden ihn durch Aufruf von Makros aus der Methodenbank. Letztere können vom Anwender modifiziert oder ersetzt werden. Ebenfalls lassen sich eigene Makros einfügen. In gleicher Weise können zu den Standardbausteinen von TESYS weitere hinzugefügt und mit Code gefüllt werden. In der Methodenbank können außer Zielsprachencode, Pseudocode, TESYS-Anweisungen auch freie Texte abgespeichert sein.

Die in TESYS vorgesehenen Dokumentationssätze lassen sich auch um anwenderspezifische erweitern. Abgesehen von den Prozessoren des Dokumentationssystems erzeugen alle Prozessoren Zielsprachencode, Dokumentationssätze und aufbereitete Listen.

## 4. Vorteile der TESYS-Anwendung

Die große Mächtigkeit der TESYS-Anweisungen und die Nutzung änderbarer Standardparameter ermöglicht einen geringen Umfang der vom Anwender vorzunehmenden Spezifikation. Alle ableitbaren Angaben werden von den Prozessoren erzeugt. Die Integration der Prozessoren gewährleistet die Paßfähigkeit untereinander und eine hohe Zuverlässigkeit. Das Bausteinprinzip ermöglicht eine funktionsorientierte Betrachtungsweise über den gesamten Entwicklungszeitraum, also auch in der Implementierungsphase, weil funktionell zusammengehörender Code unabhängig von der erforderlichen Stellung im Programm notiert werden kann und TESYS die Umordnung mit Hilfe der Bausteinnamen vornimmt. Außerdem wird durch die Methodenbank die Wiederverwendung von Lösungen im Sinne von Standardbausteinen gefördert. Die Mittel von TESYS gestatten auch eine standardisierte Anwendung von Vorgehensweisen, Namen u. a. vorzugeben und zu kontrollieren.

Verfasser:
Dipl.-Ing. Ernst Nikulski
VEB Robotron Zentrum für
Forschung und Technik
8012 Dresden
Leningrader Str. 15

Bertholdt, Dieter

## Ein allgemeines Steuerprogramm zur sequentiellen Auswertung von Dateien

Aufgabenstellung und Anliegen des Programms:
Die sequentielle Auswertung von Dateien ist eine typische Aufgabe für Projekte der Planung, Leitung und Kontrolle. Im Verlaufe der Lösung solcher Aufgaben werden eine oder mehrere Dateien sequentiell gelesen, wobei das Lesen der Sätze dieser Dateien anhand eines Ordnungsbegriffs, der diesen Dateien gemeinsam ist, und nach dem die Dateien sortiert sind, synchronisiert wird. Die Daten der Sätze mit gleichem Ordnungsbegriff werden miteinander verknüpft und auf eine oder mehrere Dateien, ebenfalls sequentiell, geschrieben. Auch den im Speicher der EDVA auftretenden Zwischenergebnissen läßt sich ein entsprechender Ordnungsbegriff zuweisen. Die Reihenfolge der Verarbeitung der einzelnen Sätze ist durch eine aufsteigende Folge dieses Ordnungsbegriffs gegeben. Insgesamt wird der Verarbeitungsprozeß durch die im Verlaufe der Bearbeitung auftretenden Dateneinheiten (Strukturen) und ihre zugehörigen Ordnungsbegriffe bestimmt. Die Verarbeitung läßt sich in eine gewisse Zahl von Teilprozessen auflösen, die durch entsprechende Teilprogramme (Module) realisiert werden. Jeder dieser Module benötigt jeweils ganz bestimmte Strukturen als Datenquellen, beziehungsweise Datensenken. Die Koordinierung der Teilprozesse in Form eines Ansteuerns der verschiedenen Module läßt sich durch ein allgemeines Steuerprogramm durchführen, das unabhängig von den speziell zu lösenden Aufgaben ist. Die Module werden von steuernden Programmfunktionen weitgehend entlastet. Die verschiedenen Module werden vom Steuerprogramm als unabhängig voneinander betrachtet. Sie stehen lediglich über ihre Ein- und Ausgabestrukturen miteinander in Verbindung.

Steuerblöcke:
Das Steuerprogramm unterscheidet 3 Typen von Objekten: Strukturen, Module, Dateien. Jedes dieser Objekte wird durch einen Steuerblock charakterisiert.

Struktur:
Das Steuerprogramm erhält mit Hilfe des Steuerblocks Namen und Adresse der Struktur, eine Vorschrift zur Bildung des zugehörigen Ordnungsbegriffs und eine Liste spezieller Module, die der Initialisierung oder Finalbearbeitung der Struktur dienen. Vom Steuerprogramm wird ein Zähler (Existenz) bereitgestellt, der die Zahl der Module angibt, die diese Struktur als Ausgabestruktur benutzen und zum betreffenden Ordnungsbegriff auch tatsächlich die Steuerung hatten. Der Zähler ist zur Kommunikation des Steuerprogramms mit den Modulen oder der Module untereinander gedacht. Das Steuerprogramm benutzt für seine eigenen Zwecke zwei weitere Zähler, um die Freigabe oder Blockierung der Strukturen für Ein- oder Ausgabeoperationen der Module zu regeln.

Modul:
Das Steuerprogramm erhält mit Hilfe des Steuerblocks Namen und Adresse des Moduls, eine Liste der Eingabestrukturen und eine Liste der Ausgabestrukturen des Moduls. Vom Steuerprogramm wird ein Zähler (Existenz) bereitgestellt, der die Zahl der zum aktuellen Ordnungsbegriff tatsächlich existierenden Eingabestrukturen des Moduls (d.h. Strukturen mit einem Existenzzähler größer als Null) angibt. Auch mit Hilfe dieses Zählers kann eine Kommunikation des Steuerprogramms mit den Modulen oder der Module untereinander erfolgen.

Datei:
Dem Steuerprogramm werden mit Hilfe dieses Steuerblocks Namen (DD-Name), Steuerblockadresse (DCB) und Typ der Datei, sowie die zugehörigen Ein- oder Ausgabestrukturen mitgeteilt.

Realisierung:
Das Programm ist für ESER-Anlagen unter Steuerung des Betriebssystems OS bestimmt. Die Angaben zu den Steuerblöcken werden in Form von Assembler-Makros gemacht. Module werden in PL1 formuliert. Sie beginnen mit dem Modulnamen als Marke und enden mit einem Sprungbefehl unter Benutzung einer allen Modulen gemeinsamen Label-Variablen.
Mit Hilfe der Angaben zu den Steuerblöcken und des PL1-Quelltextes der Module wird ein kompletter PL1-Programmtext erzeugt, der neben den Algorithmen der Module und den Deklarationen der vom

Steuerprogramm zu verwaltenden Strukturen, die Deklarationen der Steuerblöcke enthält. PL1-Compiler und Programmverbinder erzeugen aus diesem Quelltext das Verarbeitungsprogramm, das in der nächsten Phase, der eigentlichen Aufgabenlösung, zur Ausführung gelangt.

Programmablauf:
Das Verarbeitungsprogramm erhält die Steuerung, initialisiert die Steuerblöcke und stellt die für alle zur Abarbeitung gelangenden Module gleiche PL1-Umgebung her. Danach wird die Steuerung an das Steuerprogramm abgegeben. Das Steuerprogramm fixiert die PL1-Umgebung und koordiniert mit Hilfe der Steuerblöcke und verschiedener Wartelisten die Zusammenarbeit der Module. Ein Modul wird angesteuert, falls alle seine Eingabestrukturen für die Eingabe und alle seine Ausgabestrukturen für die Ausgabe frei sind. Dieser Zustand wird durch einen Wert größer als Null für die betreffenden Zähler angezeigt. Umgekehrt haben dann die Zähler für Ausgabe und für Eingabe dieser Strukturen einen Wert gleich Null. Bevor einem Modul die Steuerung übergeben wird, stellt das Steuerprogramm die vom Modul benötigte PL1-Umgebung her und weist der Labelvariablen, mit deren Hilfe der Rücksprung aus dem Modul erfolgt, einen dem Steuerprogramm entsprechenden Wert zu. Nach Rückgabe der Steuerung durch den Modul werden entsprechend den erreichten Werten der Ordnungsbegriffe, die Zähler für Eingabe und Ausgabe der vom Modul benutzten Ein- und Ausgabestrukturen um Eins erniedrigt. Erreicht ein Zähler den Wert Null, werden die zur Struktur gehörigen Initial- oder Finalmodule angesteuert. Der Verarbeitungsprozeß endet, wenn für alle Eingabedateien die Dateiendebedingung anliegt und allen internen Datenstrukturen der Ordnungsbegriff 'Ende' zugewiesen wurde. Das Steuerprogramm gibt dann die Steuerung an das Verarbeitungsprogramm zurück und dieses beendet seine Arbeit durch Rückgabe der Steuerung an das Betriebssystem.

Verfasser:
Dipl.-Math. Dieter Bertholdt
Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum
1086 Berlin
Unter den Linden 6

Bauer, Günther

## Programmierung und Nachnutzbarkeit

Die Vorteile der Wiederverwendbarkeit von POS leuchten intuitiv unmittelbar ein, warum aber werden sie so wenig genutzt? Welche objektiven Gründe gibt es für die viel zu geringe Nachnutzung auf diesem Gebiet?

Die Wiederverwendung von POS ist mit zwei grundlegenden Bedingungen verbunden:

1. Gleichartige DV-Aufgaben müssen wiederholt auftreten.
2. Nachnutzung muß ökonomisch vorteilhaft sein.

Die erste Bedingung kann als notwendige, die zweite als hinreichende angesehen werden.

Wie steht es nun um die Erfüllung dieser Voraussetzungen? Das mehrfache Vorkommen gleichartiger DV-Aufgaben läßt sich beinahe täglich in der Praxis der Herstellung von Systemunterlagen (Programmierung) sowohl im Kleinen als auch in größerem Rahmen beobachten. Erinnert sei z.B. daran, daß unzählige Institutionen ihr eigenes Programmsystem zur Ermittlung von Lohn und Gehalt erarbeitet haben.

Zu den ökonomischen Verhältnissen im Zusammenhang mit Softwareherstellung sind in letzter Zeit zahlreiche Fakten zusammengetragen worden. Betrachtet man sie unter dem Blickwinkel der Nachnutzung, ergeben sich zusätzliche Erkenntnisse.

Die Kosten für eine Million Maschinenoperationen haben sich nach Hansen (1979) wie folgt entwickelt:

| | |
|---|---|
| 1950 | 30,00 DM |
| 1970 | 0,30 DM |
| 1979 | 0,03 DM. |

Verbunden mit einer analogen Entwicklung der Kosten pro Speicherplatz vollzog sich die drastische Veränderung des so häufig zitierten Aufwandsverhältnisses von Hardware und Software. Es zeigt sich also, daß die insgesamt größeren Anteile zur Rationalisierung im Softwarebereich liegen.

Detaillierteren Aufschluß über die Möglichkeiten der Nachnutzung geben Aussagen über den Herstellungsprozeß. Abbildung 1 ist eine Zusammenstellung statistischer Werte zur Fehlerentstehung und -erkennung während der Erarbeitung und des Einsatzes von Software (vgl. Ross 1976).

| Fehlerent-stehung / Fehler-erkennung | Entwurf [%] | Codierung und Test [%] | Summe [%] |
|---|---|---|---|
| beim Test | 19 | 27 | 46 |
| während und nach der Abnahme | 45 | 9 | 54 |
| Summe | 64 | 36 | |

Abb. 1 Fehlerentstehung und -erkennung

Die Mehrzahl der Fehler (64 %) bei der Herstellung von Systemunterlagen wird der Statistik zufolge in den Phasen vor der Codierung verursacht.

Das belegt, daß der Entwurf von Programmsystemen der schwierigere Teil des Gesamtprozesses ist und die sich dabei einschleichenden Unkorrektheiten besonders unangenehme Auswirkungen haben. Eine günstige Beeinflussung dieser Etappen auch durch Nachnutzung ist demnach besonders wünschenswert. Der Aufwand für die Beseitigung der Fehler, die während Codierung und Test verursacht werden, kann bei der Nachnutzung im allgemeinen gänzlich entfallen. Wie hoch dieser Anteil am Gesamtaufwand ist, zeigt die Abbildung 2 - Aufwandsverteilung bei der Herstellung von Systemunterlagen (vgl. Keßler 1981).

| Phase | Aufwand [%] |
|---|---|
| Entwurf | 20 |
| Codierung | 10 |
| Test | 30 |
| Industrie-erprobung | 20 |
| Korrektur + Test | 20 |

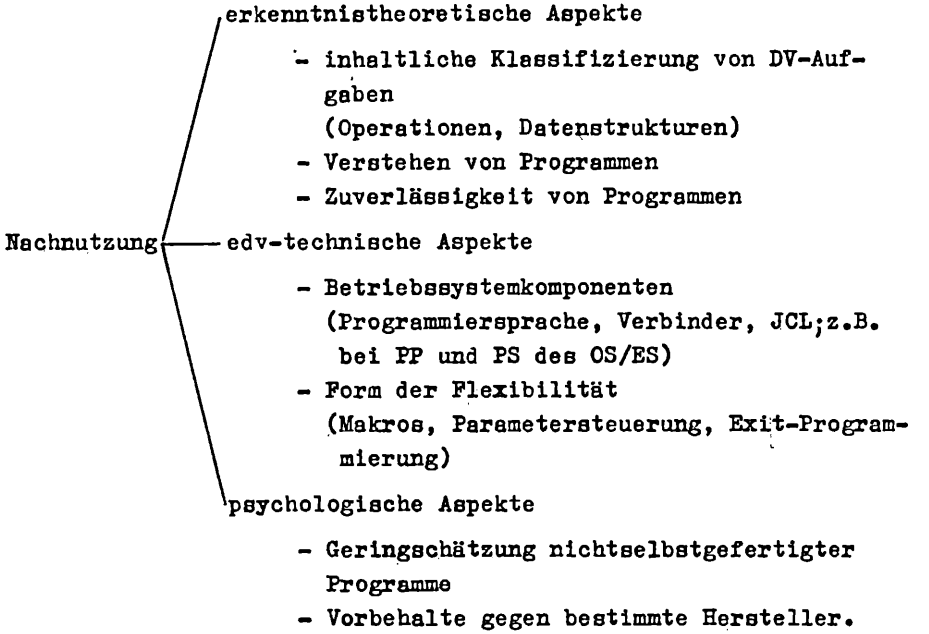
Abb. 2 Aufwandsverteilung

Somit wird deutlich, daß die Verhältnisse hinsichtlich der beiden grundlegenden Bedingungen für die Nachnutzung günstig sind; der erreichte Faktor der wiederholten Verwendung aber ist nicht ausreichend. Diese Situation ist auf zwei wesentliche Ursachen zurückzuführen:

1. Die Vielschichtigkeit der Nachnutzungsproblematik.
2. Die fast ausschließliche Orientierung auf Endprodukte der Programmierung.

Eine eingehende Betrachtung der Problemstellung Nachnutzung führt zu drei bestimmenden Bereichen, die sich zum Teil stark voneinander unterscheiden und von denen jeder weiter untergliedert werden kann.

Die Nachnutzung vorhandener Programme setzt voraus, daß deren Funktionen verstanden werden. Dieses Verstehen an Hand der Programme und deren Beschreibung als Endprudokte zu erreichen, erweist sich in der Regel als sehr schwierig. Das bedeutet großen Aufwand.




Wenn es gelingt, den Prozeß der vorgenommenen Umformung einer Aufgabe in Programme über markante Zwischenstationen nachvollziehbar zu machen, kann das Verstehen entscheidend erleichtert werden. Für das Auffinden geeigneter Zwischenstationen ist dieser Prozeß der Überführung einer EDV-Aufgabe in Programme zu analysieren. Allgemein üblich ist es dabei, eine Einteilung in Phasen vorzunehmen; die einzelnen Ansätze unterscheiden sich in der Zahl der Teilabschnitte und ihren Bezeichnungen.

Programmieren umfaßt im allgemeinen die Strukturierung von Funktionen und von Daten. Für die Strukturierung von ausschließlich Daten hat sich das Dreiebenenmodell bewährt. Das legt es nahe, dieses Modell auf die Programmierung als Gesamtheit anzuwenden. Die Verknüpfung von Phaseneinteilung und Dreiebenenmodell ist in Abbildung 3 vorgenommen worden.

Das Dreiebenenmodell macht besonders deutlich, daß in den einzelnen Programmierphasen unterschiedliche Produkte mit unterschiedlichen Ausdrucksmitteln entstehen, die in unterschiedlichem Maße zwecks Nachnutzung von Programmen oder auch von Zwischenprodukten das Verstehen

unterstützen können. Die Abschnitte der Programmierung, die zwischen konzeptueller und interner Ebene liegen, werden gegenwärtig zunehmend automatisiert (Programmentwicklungssysteme).

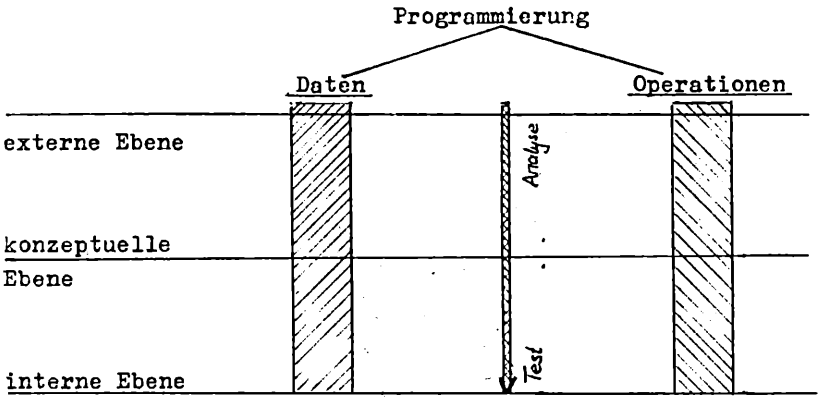



Abb. 3 Verknüpfung von Phasen- und Dreiebenenmodell

Für die Nachnutzung von POS vorteilhaft können in besonderem Maße die Produkte sein, die im Bereich zwischen externer und konzeptueller Ebene erstellt werden. Als Beispiel dafür wird kurz auf die in Hansen (1979) vorgeschlagene problemorientierte Kriterienliste eingegangen.

Abbildung 4 zeigt einen Auszug einer solchen Liste für die Aufgabe Kostenrechnung. Diese Darstellung einer DV-Aufgabe und ihrer Teilaufgaben ermöglicht die Kommunikation unterschiedlicher Diskussionspartner, ohne spezifische DV-Kenntnisse bzw. Implementierungscharakteristika voraussetzen zu müssen. In der am weitesten rechts angeordneten Spalte wird für ein konkretes System angezeigt, welche Teilaufgabe realisiert ist bzw. welche Funktionen von einer einzurichtenden EDV-Anwendung übernommen werden sollen. Derartige Kriterienlisten können das Ergebnis der Analyse einer Aufgabenstellung zur Herstellung bzw. Anwendung von Systemunterlagen sein. Möglich sind dabei Untersuchungen im entsprechenden Fachbereich, ohne in diesem Stadium durch EDV-Spezifika belastet zu werden.

| Kostenarten-rechnung | | | |
|---|---|---|---|
| Eingabe | 1. Kostenübernahme | aus | Finanzbuchhaltung |
| | | | anderem Inf.-System |
| | | von | Belegen |
| | 2. Maximale Anzahl | der | Kostenarten |
| | | | Mengenangaben |
| | | | Zeitangaben |
| Verarbeitung | 1. Prüfung auf ~~der~~ Doppelbelege | | |
| | 2. Primärkostenverbuchung | auf | Kostenstellen |
| | | | Kostenträger |
| | | | Aufträge |
| | | | Ergebnisbereiche |
| Ausgabe | 1. Abstimmlisten | für | Kostenarten |
| | | | Erlösarten |

Abb. 4 Problemorientierte Kriterienliste

Weiterhin kann eine so aufgebaute Aufgabenbeschreibung als Ausgangspunkt für den Übergang vom WAS zum WIE angesehen werden. Diese Kriterienliste ist hierarchisch aufgebaut und läßt sich deshalb leicht als Struktur darstellen, wie sie in HIPO oder auch bei Jackson (1975) benutzt wird. Damit läßt sich demzufolge der Entwurf entsprechender Systemunterlagen schrittweise dokumentieren. Diese Dokumentation kann dann bis zum gewünschten Verfeinerungsgrad verfolgt werden, um z.B. erforderliche Anpassungen zum Zwecke der Nachnutzung vornehmen zu können.

Zusammenfassend wird festgestellt, daß aufgrund der Vielschichtigkeit der Problematik Nachnutzung eine spürbare Verbesserung der Situation nur durch ein ganzes Bündel von Maßnahmen erreicht

werden kann. Dabei ist es notwendig, daß entsprechend den jeweiligen Gegebenheiten an der Weiterentwicklung der einen oder anderen Teilproblematik gearbeitet wird. Es müssen jedoch Aktivitäten solcher Art in den Gesamtzusammenhang der Nachnutzungsaspekte eingeordnet werden. Die Überbetonung einer Seite der Wiederverwendung bei Vernachlässigung der noch zahlreichen anderen führt zu unbefriedigenden Ergebnissen. Die zielgerichtete Aufbereitung von Zwischenprodukten bei der Herstellung von Systemunterlagen kann eine Komponente zur Ausdehnung der Nachnutzung sein.

Literaturhinweise

/1/ Hansen, H.R.: Möglichkeiten und Probleme beim Einsatz von Standard-Anwendungssoftware. IBM Nachrichten 29(1979),25 S. 149 - 155

/2/ Jackson, M.: Principles of Program Design. London 1975

/3/ Keßler, G.: Anwendung der Rechentechnik zur Intensivierung von Forschungs- und Entwicklungsprozessen. rd/Beiheft 1/1981 S. 2 - 8

/4/ Ross, D.: Homilies for Humble Standards. CACM 19(1976),11 S. 39 - 42

Verfasser:
Dr. Günther Bauer
Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum
1086 Berlin
PSF 1297

Kiesohke, Reiner

# Auswahlkriterien für die Anwendung vorgefertigter Software der Dialogverarbeitung

## 1. Einleitung

Die Erarbeitung von dialogorientierten Anwendungslösungen kann durch die Nutzung von vorgefertigter Software der Dialogverarbeitung unterstützt werden. Im Vortrag wurde dargelegt, wie diese Software in Abhängigkeit von den in ihr implizierten Serviceleistungen pro zu durchlaufender Programmschicht klassifiziert werden kann. Grundsätzlich sind bei der Abarbeitung eines Programmsystems der Dialogverarbeitung folgende Programmschichten zu durchlaufen:

- die Leitungssteuerung,
- die Nachrichtensteuerung und
- die Nachrichtenverarbeitung.

Gleichzeitig wurde im Vortrag aufgezeigt, unter welchen Bedingungen die so klassifizierte vorgefertigte Software der Dialogverarbeitung genutzt werden sollte.

## 2. Im Data Management des Betriebssystems implizierte Komponenten

Für die Betriebssysteme ist charakteristisch, daß in ihnen zunehmend Serviceleistungen zur Unterstützung der Dialogverarbeitung enthalten sind (vgl. z. B. /1/, /2/). Insbesondere das Data Management enthält Module, die für die Anwendung des Mensch-Maschine-Dialoges (MMD) genutzt werden können. Das Data Management unterstützt die unterste Programmschicht, die bei der Realisierung des MMD zu durchlaufen ist - die Leitungssteuerung. Diese Softwareschicht bildet eine unmittelbare Schnittstelle mit der Hardware einer EDVA.

Die Softwarekomponenten, die unter den Bedingungen der lokalen Datenverarbeitung (DV) zur Leitungssteuerung gehören, realisieren datenverarbeitungsspezifische Funktionen, die für das Übertragen von Daten vom Terminal zur Zentraleinheit der EDVA und in umgekehrter Richtung notwendig sind, z. B.

- Abrufen der Daten vom Terminal und Übermittlung dieser Daten

zur EDVA;
- Zustandsüberprüfung des Terminals und der Übersetzungsleitungen;
- Einleitung von Fehlermaßnahmen bei fehlerhafter Datenübertragung zwischen dem Terminal und der EDVA;
- Übergabe der Ausgabedaten eines Nachrichtenverarbeitungsprogramms (Problemprogramm) an die Ausgaberoutinen der Datenverwaltung zur Übermittlung an das adressierte Terminal;
- Weiterleitung der Eingabedaten vom Terminal an das entsprechende Nachrichtenverarbeitungsprogramm;
- Verwaltung von Warteschlangen bezüglich der Ein- und/oder Ausgabedaten und der benötigten Nachrichtenverarbeitungsprogramme.

Das Anfordern von Serviceleistungen des Betriebssystems für die Unterstützung des MMD unter lokalen Bedingungen setzt detaillierte Programmierkenntnisse voraus, in der Regel in der Programmiersprache, auf deren Grundlage das betreffende Betriebssystem programmiert wurde. Unter den Bedingungen der Anwendung von ESER-Rechnern müssen Programme für die Dialogverarbeitung beispielsweise in ASSEMBLER programmiert werden, um die Leistungen des Betriebssystems für die Realisierung des MMD nutzen zu können. Hierfür sind spezielle imperative und deklarierende Makros zu verwenden und die Zustände von Bits in Steuerblöcken abzutesten, die vom Betriebssystem in den Programmen für den MMD definiert werden.

Soll die Realisierung des MMD sowohl unter den Bedingungen der lokalen Datenverarbeitung als auch der Datenfernverarbeitung (DFV) möglich sein, können ebenfalls Serviceleistungen des Betriebssystems für die DFV bezüglich der Datenverwaltung genutzt werden. Diese Leistungen werden vom Betriebssystem in Form einer "Basiszugriffsmethode" bereitgestellt (vgl. z. B. /3/).

Die Inanspruchnahme von Leistungen des Betriebssystems für die DFV ist ähnlich kompliziert wie bei der Nutzung der Unterstützung des MMD unter den Bedingungen der lokalen Datenverarbeitung.

Für die Nutzung der Serviceleistungen des Betriebssystems betreffs der Leitungssteuerung sind ein umfangreiches Detailwis-

sen und ein hoher Entwicklungsaufwand erforderlich. Aus diesen Gründen ist diese Form der Standardsoftwareunterstützung unmittelbar für die Anwendung des MMD zur Rationalisierung von Datenverarbeitungsaufgaben nicht geeignet. Hinzu kommt, daß unter diesen Bedingungen die notwendige Software für die mittlere und oberste Schicht eines Dialogprogrammsystems ebenfalls entwikkelt werden muß.

### 3. Unterstützung der Anwendung des MMD durch komplexe Modulsysteme

Komplexe Modulsysteme für die Realisierung des MMD basieren auf den Standardleistungen des Betriebssystems. Für ihren Leistungsumfang ist aber charakteristisch, daß neben der Leitungssteuerung auch die Nachrichtensteuerung unterstützt wird. Die Programme für die Nachrichtensteuerung beinhalten solche Funktionen wie

- die Initialisierung der Dialogverarbeitung am Terminal;
- die Steuerung des Datenaustausches zwischen dem Problemprogramm und dem adressierten Terminal;
- die Protokollierung und Behandlung von Fehlern, die vom Betriebssystem bei der Datenübertragung zwischen einem Terminal und der Zentraleinheit der EDVA erkannt wurden;
- die Vermeidung der Beeinflussung eines gestörten Terminals auf ordnungsgemäß arbeitende Terminals (so darf z. B. die Abschaltung eines gestörten Terminals die Arbeit an den anderen aktiven Terminals nicht behindern);
- die Realisierung von Rechnerzeitzuweisungsprinzipien für die Nachrichtenverarbeitungsprogramme (Problemprogramme).

Die Nachrichtenverarbeitung wird in der Regel in den Problemprogrammen realisiert. Während die problemspezifische Nachrichtenverarbeitung ausschließlich in den Problemprogrammen erfolgt, sind Softwarekomponenten der Nachrichtenverarbeitung, die für die multivalente Nachnutzung geeignet sind, häufig ebenfalls Bestandteile von den Modulen der Dialogverarbeitung. So sind in diesen Modulen oft folgende Funktionen der Nachrichtenverarbeitung realisiert:

- Realisierung von Komponenten des SPOOL-Prinzips (z. B. für Lese- und Schreiboperationen werden keine mechanischen Ein-

und Ausgabegeräte unmittelbar verwendet, sondern nur externe Speicher, die einen schnellen Zugriff gestatten).
- Kommunikationsmöglichkeiten zwischen den Terminalnutzern untereinander und/oder zwischen ihnen und dem Operator an der EDVA.
- Steuerung der terminalbezogenen Protokollierung von Ausgabeinformationen über den Systemdrucker der EDVA, die der Nutzer am Terminal wahlweise abfordert.
- Wahlweise Protokollierung der Terminalsitzung, wenn es der Nutzer wünscht, in einer sogenannten LOG-Datei auf einem externen Speicher (diese Funktion ist notwendig, wenn die Gefahr eines Systemausfalls besteht und unter Umständen eine RESTART-Organisation notwendig werden kann).
- Steuerung der Kompatibilität zwischen den im Problemprogramm und in den Modulen für die Unterstützung der Anwendung der Dialogverarbeitung verwendeten Programmiersprachen.
- Unterstützung der Arbeit mit Direktzugriffsdateien im Nachrichtenverarbeitungsprogramm.

In einem Modulsystem für die Unterstützung der Dialogverarbeitung müssen neben den Softwarekomponenten für die Leitungs- und Nachrichtensteuerung zusätzlich Leistungen enthalten sein, die das Arbeiten mit Dateien im Direktzugriff im Problemprogramm einfach gestalten. Die Anwendung des MMD für die Rationalisierung von Datenverarbeitungsaufgaben ist ohne die Möglichkeiten des Direktzugriffs zu den benötigten Dateien nicht akzeptabel, insbesondere zur Sicherung des angestrebten Antwortzeitverhaltens zwischen einem Nutzer und einem Rechnersystem. Die Inanspruchnahme der Möglichkeiten des Direktzugriffs zu Dateien im Rahmen der konventionellen Programmiersprachen (z. B. in den Programmiersprachen ASSEMBLER oder PL/1) ist kompliziert, insbesondere wenn eine Datei geladen werden soll. Aus diesem Grunde müssen zum Modulsystem für die Unterstützung der Dialogverarbeitung Programme gehören, die mittels der Makro- und/oder Prozedurtechnik für das Laden und/oder das Lesen und/oder das Rückschreiben von Datensätzen einfach im Problemprogramm zu nutzen sind. Außerdem sollte zum Modulsystem für die Unterstützung der Dialogverarbeitung ein Dateimanipulationssystem für die Wartung der Direktzugriffsdateien gehören. Dieses Dateimanipulationssystem muß

zumindest die "Update"-Funktionen (Insert- und/oder Replacement- und/oder Delete-Funktion für einen Datensatz in einer Datei) und das Lesen einer Datei und/oder eines Datensatzes der Datei am Terminal ermöglichen.

Komplexe Modulsysteme sind für die Unterstützung der Anwendung der Dialogverarbeitung bei der Rationalisierung geeigneter Datenverarbeitungsaufgaben nutzbar und sollten vorrangig dann angewendet werden, wenn abarbeitungsbereite Standard-Programmsysteme der Dialogverarbeitung für das eingesetzte Rechnersystem nicht verfügbar sind. Aber auch beim Vorhandensein der nachfolgenden Bedingungen ist der Einsatz komplexer Modulsysteme akzeptabel:

- die Anwendung des MMD erfolgt nur durch solche Nutzer, die nicht programmieren wollen (Dialogcompiler und -interpreter werden von solchen Nutzern nicht benötigt);
- die Anwendung des MMD setzt keine Datenbank voraus;
- die zur Verfügung stehende Kapazität des Hauptspeichers und der externen Speichermedien, die den schnellen und wahlfreien Zugriff gestatten, ist begrenzt;
- die Rationalisierung des Modulsystems soll unter der Verantwortung des Nutzers möglich sein (es ist schwierig, in die Programmstruktur abarbeitungsbereiter Standard-Programmsysteme der Dialogverarbeitung einzudringen);
- die Anwendung des MMD erfolgt nur im Teilhaberbetrieb;
- das Testen der Problemprogramme erfolgt nur im Stapelbetrieb.

## 4. Abarbeitungsbereite Standardprogrammsysteme der Dialogverarbeitung

Diese Programmsysteme unterstützen die Leitungs- und Nachrichtensteuerung sowohl unter den Bedingungen der lokalen Datenverarbeitung als auch unter den Bedingungen der Datenfernverarbeitung umfassender als komplexe Modulsysteme und enthalten auch für die Nachrichtenverarbeitung umfangreichere Softwarekomponenten. Der Leistungsumfang dieser Programmsysteme kann nach ihrer Generierung beim Anwender ohne Programmieraufwand sofort genutzt werden. Es ist aber erkennbar, daß die Standard-Programmsysteme, die gegenwärtig angeboten werden, für die Nachrichtenverarbeitung vorrangig nur multivalent einsetzbare Problemprogramme enthalten, z. B. Editorsysteme, mehrere Compiler, Syntaxprüfer und

Programmgeneratoren für die Unterstützung der Entwicklung von problemspezifischer Software. Die Erstellung der problemspezifischen Programmsysteme geschieht vorrangig unter der Verantwortung des Nutzers (z. B. durch Eigenprojektierung und/oder Kauf).

Abarbeitungsbereite Standard-Programmsysteme können als Teilnehmer- und/oder Teilhabersysteme genutzt werden. Während Teilnehmersysteme insbesondere die Arbeit mit Dateien unterstützen, sind Teilhabersysteme entweder vorrangig für die Arbeit mit Dateien oder für die Arbeit mit einer Datenbank ausgelegt.

Die Leistungen eines abarbeitungsbereiten Standard-Programmsystems werden durch eine Kommandosprache und/oder Makros bzw. Prozeduren in den problemspezifischen Programmen in Anspruch genommen.

Abarbeitungsbereite Standard-Programme sind vor allem dann zu nutzen, wenn folgende grundsätzliche Bedingungen für die Rationalisierung von Datenverarbeitungsaufgaben vorliegen:

- umfangreiche Serviceleistungen der Dialogverarbeitung für die Erarbeitung von Problemprogrammen sind erforderlich (z. B. wenn viele Problemprogramme zu erarbeiten sind);
- die umfassende Nutzung der Unterstützung des Betriebssystems für die Arbeit mit Dateien ist am Terminal notwendig;
- der Einsatz einer Datenbank ist geplant bzw. bereits unter Bedingungen der Stapelverarbeitung realisiert;
- die Probleme des Mangels an Hauptspeicher- und externer Speicherkapazität sowie der Realisierung einer hohen Operationsgeschwindigkeit bei dem genutzten Rechnersystem sind weitestgehend irrelevant;
- die Anwendung des MMD ist in Teilnehmer- und Teilhabersystemen vorgesehen;
- die Anwendung des MMD ist für viele unterschiedliche Nutzer mit sehr differenziertem Aufgabenspektrum konzipiert;
- die Anwendung des MMD wird sowohl unter den Bedingungen der lokalen DV als auch der DFV realisiert.

Die unmittelbare Anwendung abarbeitungsbereiter Standard-Programmsysteme für die Realisierung des MMD erfordert gegenwärtig umfassende Kenntnisse vom benutzten Betriebssystem und zum Teil detaillierte Programmierkenntnisse. Aus diesem Grunde sollte

bei der Nutzung von abarbeitungsbereiten Standard-Programmsystemen für den MMD die Programmierung (einschließlich Test) der benötigten Problemprogramme von Spezialisten der DV durchgeführt werden.

Literaturhinweise

/1/ Haupt, D. u. a., Betriebssystem OS/ES, Jobsteuerung/Ein- und Ausgabe, BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1977

/2/ Makros für die Bildschirmunterstützung ESER/ROBOTRON 21 (DOS/ES), Systemunterlagendokumentation C 4453-2458-1, VEB Kombinat Robotron, Zentralvertrieb, Dresden 1973

/3/ Autorenkollektiv, Datenfernverarbeitung (Systemunterlagen und Anwendungen), Rechentechnik/Datenverarbeitung 15(1978), 2. Beiheft

/4/ Bode, G., Neue Methoden und Techniken der Programmierung, Teil 2: Dialogprogrammierung auf IBM-Systemen, IBM-Nachrichten 25(1975), Heft 223

/5/ Fischer, U. E., TSO - universeller Teilnehmerbetrieb auf IBM-Systemen /370 und /380, Sonderdruck aus IBM-Nachrichten, Heft 209/1972

/6/ Lampenscherf, S. u. a., System virtueller Maschinen (SVM/ES), Rechentechnik/Datenverarbeitung 18(1981)2

/7/ Dialogsystem DLS/ES - Nutzerhandbuch, Projektierungshandbuch, Kommandohandbuch, VEB Leitzentrum für Anwendungsforschung der VVB Maschinelles Rechnen, Berlin 1979

Verfasser:

Dr. sc. oec. Reiner Kieschke

Hfö Berlin

Sektion Leitung, Informationsverarbeitung und Statistik

Wissenschaftsbereich Datenverarbeitung

1157 Berlin

Hermann-Duncker-Str. 8

Meuche, Hans-Friedrich

Wegener, Edith

# Anwendung des Mensch-Maschine-Dialoges für betriebwirtschaftliche Aufgaben

## Vorbemerkungen - Standpunkt

In hochindustrialisierten Ländern wird der Anwendung des Mensch-Maschine-Dialoges als einer Betriebsform der Nutzung elektronischer Rechentechnik viel Aufmerksamkeit gewidmet. Immer neue Anwendungsgebiete werden für den Mensch-Maschine-Dialog erschlossen. Diesen Trend verdeutlicht nachstehende Gegenüberstellung.

| Betriebsform | 1971 | 1981 | 1985/1990 |
|---|---|---|---|
| Stapelbetrieb | 97 | 57 | 26 |
| Dialogbetrieb | 3 | 43 | 74 |

Verhältnis von Stapel- zu Dialogbetrieb in der Anwendung (Angaben in Prozent) nach: /6/

Diese Entwicklung muß bei der Projektierung von DV-Systemen stärker als bisher beachtet werden, wobei zwei Tendenzen aus betriebswirtschaftlicher Sicht besonders hervorzuheben sind:

1. Der Betriebswirtschaftler muß weit mehr als bisher die Datenverarbeitungsprojektierung direkt beeinflussen und die Gestaltung interaktiver Systeme inspirieren.
   Besonders notwendig ist die intensivere Arbeit des Betriebswirtschaftlers in der Datenverarbeitungsprojektierungs-Phase "Systementwurf", um bereits hier die wichtigsten Komponenten seines Fachgebietes berücksichtigen zu können und um häufig auftretende Fehler auszuschalten. Dabei darf er sich nicht auf die Einflußnahme auf das Organisationsprojekt beschränken.
2. Der Betriebswirtschaftler muß die Dialogführung im Rahmen interaktiver Systeme in die Lösung seiner täglichen Arbeitsaufgaben integrieren.
   In fortgeschrittenen Betrieben und Kombinaten werden mit Hilfe

des Mensch-Maschine-Dialoges bereits zahlreiche betriebswirtschaftliche Arbeitsaufgaben rationalisiert. Als Beispiele seien genannt: die Materialdisposition, die Materialbestandsführung, die kurzfristige Planung, Lenkung und Kontrolle der Produktion, die Kontenführung im Kontokorrent, die Terminkontrolle, die Absatztätigkeit.

Zu berücksichtigen ist in diesem Zusammenhang, daß betriebswirtschaftliche (informationelle) Prozesse elementarisiert werden müssen, um automatisierungswürdige Teilaufgaben festzustellen und aus diesen die dialogwürdigen zu bestimmen. Diese Elementarisierung sowie die Algorithmierung automatisierungswürdiger Prozesse als eine Grundvoraussetzung zu ihrer Automatisierung erfordert hohen Aufwand an lebendiger Arbeit und kann nur in relativ langen Zeiträumen realisiert werden.

Arbeitsteilung zwischen Mensch und Automat

Beim Mensch-Maschine-Dialog können den Automaten und den Menschen, entsprechend den Fähigkeiten der Dialogpartner die jeweils am effektivsten zu lösenden Aufgaben übertragen werden. Diese Arbeitsteilung zwischen Mensch und Automat ermöglicht wesentlich höhere Rationalisierungseffekte, als sie bei konventioneller Stapelverarbeitung zu erreichen sind. Um diese höhere Rationalisierungseffekte realisieren zu können, ist unter anderem auf folgende Grundüberlegungen zu achten:

- Die Arbeitsgeschwindigkeit des Dialogpartners - Mensch - ist im Verhältnis zur Arbeitsgeschwindigkeit des Dialogpartners - Automat - gering.
- Der Dialogpartner - Automat - ist in der Lage, mit einer größeren Anzahl von Dialogpartnern - Mensch - unterschiedliche oder gleiche Probleme zur gleichen Zeit zu bearbeiten.
- Die Fertigkeiten und Fähigkeiten sowie der Kenntnisstand des Dialogpartners - Mensch - sind sehr unterschiedlich und nur relativ langfristig veränderbar.
- Die Sprache des Dialogpartners - Mensch - ist nicht eindeutig.
- Das Verhalten des Dialogpartners - Mensch - ist nicht immer

streng rational und oft von physischen und psychischen Einflüssen geprägt sowie von der Umwelt abhängig.
- Die Reaktionsfähigkeit des Dialogpartners - Mensch - ist der Reaktionsfähigkeit des Dialogpartners - Automat - unterlegen.
- Oft besitzen die Endnutzer in betriebswirtschaftlichen Fachbereichen Vorbehalte oder Vorurteile gegenüber der elektronischen Rechentechnik.
- Die Fähigkeiten des Dialogpartners - Mensch - sind gegenüber dem Dialogpartner - Automat - vor allem im Bereich der Informationsspeicherung und im numerischen Bereich wenig entwickelt.
- Im Urteilsvermögen und im assoziativen Denken, im Entwickeln von Initiativen und im Erkennen von Zusammenhängen ist der Dialogpartner - Mensch - dem Dialogpartner - Automat - weit überlegen.
- Der Dialogpartner - Automat - kann die Arbeitsweise des Dialogpartners - Mensch - in bezug auf ihre Exaktheit unmittelbar kontrollieren.
- Der Dialogpartner - Automat - ist dem Dialogpartner - Mensch - in bezug auf Dauerbelastbarkeit in einem interaktiven System überlegen.

Bei Beachtung dieser Grundüberlegungen ergibt sich eine logische Arbeitsteilung im Mensch-Maschine-Dialog.
So sind beispielsweise routinemäßige, arithmetische und logische Operationen sowie die Informationsspeicherung dem Automaten zuzuordnen. Die Ansprüche an den Dialogpartner - Mensch - sind gering zu halten hinsichtlich
- Sprache (zu orientieren ist auf die Umgangssprache sowie die entsprechenden Fachtermini einschließlich einer einfachen Kommandosprache)
- Fertigkeiten und Fähigkeiten (zu orientieren ist beispielsweise auf eine einfache Bedienung der Dialoggeräte am Arbeitsplatz)
- Rationalität (beispielsweise muß unlogisches Denken und unmotiviertes Handeln des Dialogpartners - Mensch - berücksichtigt werden)
- Kenntnisstand (zu berücksichtigen ist der Kenntnisstand von

Anfängern und von Fortgeschrittenen, z. B. durch Anwendung der Menütechnik).

Damit sind grundlegende Aufgaben bei der Projektierung interaktiver Systeme verbunden, deren Erfüllung nicht vernachlässigt werden darf. Dies gilt insbesondere für Datenverarbeitungsprojekte des Mensch-Maschine-Dialoges zur Lösung betriebswirtschaftlicher Aufgaben.

Betriebswirtschaftliche Aufgaben sind unter anderem gekennzeichnet durch folgende Aspekte:

1. Sie erfordern meist große Datenbestände, um die jeweiligen Aufgaben lösen zu können, z. B. Daten aus dem Produktionsprozeß, aus dem Bereich der Materialwirtschaft, aus dem Gebiet der Lohnrechnung oder aus dem Gebiet der Grundmittelrechnung.
2. Arithmetische und logische Operationen treten massenhaft auf. Arithmetische Operationen sind jedoch im wesentlichen auf die vier Grundrechenarten beschränkt.
3. Die Sprache des Betriebswirtschaftlers ist wenig eindeutig. Untersuchungen der Zentralstelle für Primärdokumentation in Zusammenarbeit mit der Technischen Universität Dresden haben z. B. ergeben, daß im Bereich der Fertigungsorganisation für etwa 100 Informationen 1100 unterschiedliche Begriffe verwendet werden. Die theoretischen Erkenntnisse der "Fuzzi set" dürften hier ein breites Anwendungsgebiet finden können.
4. Die notwendigen Antwortzeiten bei betriebswirtschaftlichen Aufgaben zeigen eine außerordentlich große Breite. Sie liegen im Bereich von wenigen Sekunden bis zu mehreren Stunden, ggf. reichen sogar Antwortzeiten von einigen Tagen.
5. Viele betriebswirtschaftliche Aufgaben sind eingabe- und ausgabeintensiv.

<u>Effekte sind aus betriebswirtschaftlicher Sicht:</u>

- Die Möglichkeit der sofortigen Verarbeitung eingegebener Informationen führt zur Erhöhung der Aktualität gespeicherter Informationen sowie von Verarbeitungsergebnissen.
- Karteien, Drucklisten und andere Aufzeichnungen auf Papier am

Arbeitsplatz des Betriebswirtschaftlers können entfallen und lebendige Arbeit sowie Papier eingespart werden.
- Die Möglichkeit der sofortigen Kontrolle eingegebener Informationen in bezug auf Plausibilität und syntaktische Fehler führt zur Reduktion von fehlerhaften Rechnerläufen.
- Durch die Verlagerung der technischen Mittel zur Informationsverarbeitung in die Fachbereiche erfolgt eine Integration dieser Mittel in die an den Verarbeitungsergebnissen unmittelbar interessierten Struktureinheiten. Daraus resultieren unter anderem eine Erhöhung der Qualität der Arbeitsaufgaben im Fachbereich, eine Aktivierung der Nutzer der technischen Mittel im Fachbereich, eine Erhöhung der Zuverlässigkeit der Informationsverarbeitungsergebnisse.

Organisationslösungen bei Einsatz des Mensch-Maschine-Dialoges in der betrieblichen Materialwirtschaft

Aus den Aufgaben der betrieblichen Materialwirtschaft können nach bisher vorliegenden Konzeptionen und praktischen Erfahrungen zwei Komplexe genannt werden, die einer Dialogführung mit Hilfe von Bildschirmsystemen zugänglich sind und zugleich der Dialogführung bedürfen. Das sind
1. aus dem Komplex Wareneingangs- und Lieferkontrolle, Vertragserfüllungskontrolle: das operative Beschaffen von Material (im Sinne von Recherchen nach potentiellen Lieferanten) sowie das operative Mahnen von Bestellungen;
2. aus dem Komplex Bestandsführung und Disposition: das operative Freimachen von Material /vgl. 5/.

Die Einbeziehung des Mensch-Maschine-Dialoges in Organisationslösungen zur Rationalisierung der betrieblichen Materialwirtschaft erfordert unter anderem eine neue Dateikonzeption. In einem ersten Schritt ist eine geeignete Echtzeitdatei zu schaffen. In bestimmten Anwendungsfällen wird darüber hinaus eine zusätzliche Datei notwendig, die eine ungestörte Erfassung von Wareneingängen während der Warenbewegung vom Wareneingang über die Wareneingangskontrolle bis zur Materialeinlagerung ermöglicht. Eine solche Organisationsform ist erforderlich bei mehrfachen

Wareneingängen zu einer Materialart im Zeitintervall zwischen Wareneingang und Lagerübernahme. Diese "Echtzeitdatei" ist nur über die erstgenannte Echtzeitdatei ansprechbar und enthält bei Bedarf für die entsprechende Materialart der Echtzeitdatei 1 die notwendigen Erweiterungssätze.
Auf der Basis vorgenannter Dateikonzeption wurde in einem Anwendungsbetrieb der Mensch-Maschine-Dialog realisiert. Folgende Effekte wurden z. B. erzielt:

- Die Bereitstellung des Materials für Dispositionszwecke konnte etwa 10 Tage früher erfolgen als ohne Mensch-Maschine-Dialog.
- Der Mensch-Maschine-Dialog mit den Echtzeitdateien ermöglicht eine tägliche Bestandsfortschreibung.
- Operative Materialentnahmen können mit Hilfe des Mensch-Maschine-Dialoges sofort bestandswirksam gebucht werden.
- Die tägliche bestandswirksame Erfassung der Wareneingänge/Lagerübernahmen mit Hilfe des Mensch-Maschine-Dialoges ermöglichte eine 75%ige Arbeitszeiteinsparung bei den im Datenerfassungsprozeß beschäftigten Arbeitskräften.
- 123000 Lochkarten und 16000 $m^2$ Papier werden jährlich eingespart.
- Die Rückflußdauer für das Wareneingangs-/Lagerübernahmeprojekt beträgt weniger als 2,5 Jahre.

Organisationslösungen bei Einsatz des Mensch-Maschine-Dialoges zur kurzfristigen Planung, Lenkung und Kontrolle der Produktion

Die kurzfristige Planung, Lenkung und Kontrolle der Produktion ist ein sehr interessantes Anwendungsgebiet für den Mensch-Maschine-Dialog. Zahlreiche Industriebetriebe haben entsprechende DV-Projekte bereits realisiert, andere befinden sich in der Phase der Projektierung. Aus den Erfahrungen eines führenden Anwenders lassen sich folgende Aussagen abheben /vgl. 9, S. 15/16/:
Auf der Basis von fünf Bildschirmanwendungsprogrammen zur Nutzung von Echtzeitdateien (Dialogdateien) können folgende Informationen auf dem Bildschirm bereitgestellt werden:

- Informationen zum Fertigungsauftrag, z. B. Stand der Bearbeitung, Anzahl der Arbeitsgänge, Stand der Materialdeckung;

- Informationen über in Bearbeitung befindliche Fertigungsaufträge;
- Informationen zum Anarbeitungsstand eines Auftrages insgesamt bzw. nach Fertigungsbereichen.

"Weiterhin ist die tägliche Datenerfassung der Materialdeckungsmeldung am Bildschirm möglich. Sie zieht eine sofortige Aktualisierung aller Dialogdateien nach sich" /9, S. 17/.

Damit wurden z. B. folgende Effekte erzielt:
- sofortige Aktualisierung des Standes der materialgedeckten Fertigungsaufträge
- Ausdehnung des Zeitraumes für die Datenerfassung, um mit aktuellsten Informationen zentrale Rechnerläufe durchführen zu können.
- Erfassung der Materialdeckung durch den Nutzer im Fachbereich mit sofortiger Fehlerprüfung
- Erhöhung der Aktualität ausgedruckter Informationen
- Prüfung nach materialgedeckten Aufträgen gleichzeitig durch verschiedene betriebswirtschaftliche Fachbereiche.

Die Konzeption zur Weiterentwicklung der bisherigen Organisationslösung sieht unter anderem vor:
1. automatisierte Abforderung von Fertigungsbelegen über Bildschirm
2. Ausweis der Bestände der Zwischenlager in einer Bildschirmauftragsdatei
3. Automatisierung der Auftragszuweisung als eine Aufgabe der zentralen Fertigungslenkung.

Organisationslösungen bei Einsatz des Mensch-Maschine-Dialoges im Bereich Absatz

Von den im Fachbereich Absatz in Betrieben und Kombinaten der Textilindustrie zu lösenden Aufgaben können nach bisherigen Untersuchungen vor allem die Verkaufshandlungen und die Kontrollen der Verträge durch den Einsatz von Mensch-Maschine-Dialogen rationeller und effektiver gestaltet werden.

Speziell für Erzeugnisse, die mode- und saisonbedingten Schwankungen unterliegen, lassen sich durch dialoggeführte Verkaufshandlungen ökonomische Effekte erzielen. Da für diese Erzeugnisse der Bedarf qualitativ und quantitativ nur sehr grob einzuschätzen ist, treten während der Verkaufshandlungen Abweichungen zwischen dem angebotenen Produktionssortiment und den Forderungen der Käufer auf.

Während der Verkaufshandlungen sind innerhalb einer kurzen Zeitspanne zur Herbeiführung einer annähernden Interessenidentität zwischen Produzent und Käufer (Großhandel, Warenhäuser) Entscheidungen zu treffen, die die Kenntnis einer großen Informationsmenge voraussetzt.

Bei der Entscheidungsfindung ist zu berücksichtigen, daß die Forderungen der Käufer realisierungsseitig abgesichert sind und den ökonomischen Kennziffern der Betriebe und Kombinate nicht entgegenwirken. Die Unterstützung dieses Prozesses durch die Nutzung des Mensch-Maschine-Dialoges setzt u. a. eine Datenbasis mit folgendem Inhalt voraus:

- Informationen über die verfügbaren Fonds an Maschinenzeit je Maschinengruppe, an Arbeitszeit je Arbeitsgang und an Material je Materialart, untergliedert nach Zeiträumen
- Normative je Erzeugnis für Maschinenzeit, Arbeitszeit und Material je Zeitraum
- ökonomische Kennziffern des betrieblichen Reproduktionsprozesses.

Folgende Effekte lassen sich für o. g. Aufgabengebiet durch den Mensch-Maschine-Dialog erzielen:

- Abschluß von Verträgen, die den betrieblichen Ressourcen nicht widersprechen
- permanente Übersicht über verkaufsfreie Mengen je Erzeugnis und Liefertermin
- Verringerung des Datenerfassungsaufwandes durch Vermeidung von Doppelerfassung der Information durch den Betrieb und die wirtschaftsleitenden Organe
- tagfertige Vertragsabschlußkontrolle.

Literaturhinweise


/1/ Baumbach, H.-D.; Tatsch, B. — Inhalt und Logik universeller Systemelemente für den Aufbau sachgebietsorientierter Programmiersysteme am Beispiel der Planung, Lenkung und Kontrolle (PLK) der Produktion in Maschinenbaubetrieben. Dissertation A, TU Dresden, Sektion Sozialistische Betriebswirtschaft, 1975

/2/ Görlich, R. — IUS-interaktives Unterstützungssystem Nutzerhandbuch TU Dresden, Preprint 10-9-80

/3/ Heinrich, H.-J. (Herausgeber) — Computerleistung am Arbeitsplatz München, Wien, R. Oldenburg Verlag 1978

/4/ Meuche, H.-F.; Mitschke, G.; Müller, F.; Stahl, F.; Wegener, E. — Einführung in den Mensch-Maschine-Dialog aus betriebswirtschaftlicher Sicht TU Dresden, Eigenverlag, 1981

/5/ Müller, F. — Studie über den Einsatz des Mensch-Maschine-Dialoges in der Materialwirtschaft von Betrieben und Kombinaten der elektrotechnischen und elektronischen Industrie, Diplomarbeit 1980. Technische Universität Dresden, Sektion Sozialistische Betriebswirtschaft

/6/ Schmitz — Die Nutzung des ADV-Leistungspotentials in der Unternehmung, in: Angewandte Informatik, 5/1981

/7/ Steinbuch, A.P. — Organisation der Dialogdatenverarbeitung, Teil 1: Grundlagen der Dialogorganisation Bürotechnik, Baden Baden 4/1979

/8/ Steinbuch, A.P. — Organisation der Dialogdatenverarbeitung, Teil 4: Maskengestaltung Bürotechnik, Baden Baden 6/1979

/9/ Ziegler, E. — Der Einsatz des Mensch-Maschine-Dialoges bei der Kapazitätsbilanzierung zur Rationalisierung des Leitungs- und Planungsaufwandes im Bereich der kurzfristigen Planung, Lenkung und Kontrolle. Dissertation A, TU Dresden, Sektion Sozialistische Betriebswirtschaft, 1981

Verfasser: Dr. sc. oec. Hans-Friedrich Meuche
Dr. oec. Edith Wegener
Technische Universität Dresden
Sektion Sozialistische Betriebswirtschaft
8027 Dresden
Mommsenstr. 13

Kollár, Ľubor

# The Large Data Sets Processing Mathematical Software

## Abstract

Most of widely used mathematical software systems are based on the subprogramming structure, where processing data are managed as parameters of a subroutine. Using the large data sets (the internal storage is not large enough to store them), some other conceptions are needed. We prefere the use of the separate program modules connected only through external files because of the following advantages: the block concept can be well used; the variety of programming languages may be used in building and using such a system without problems in different handling of parameters - only the data representation must be in accord; standard file utilities can be easily used. The paper is devoted to some aspect of developement and implementation of the approach mentioned above.

## 1. Introduction

Our objective in this paper is to provide the brief discussion on constructing the large data sets processing mathematical software (LDS software). This software seems to be rather extensive, complex, atractive and useful piece of the program equipment. Some specific problems of developement of this system are mentioned. We will describe the module conception used to construct this software.

In section 2 the term LDS is explained and components of LDS software are defined. Some methods used in LDS software developement are presented in section 3. Section 4 contains a discussion on the adaptability and implementation problems.

## 2. Modules and Data

### 2.1. Large data Sets

Large data set (LDS) is a collection of large amount of data. But what does this "large amount" mean? Thousand or million numbers? In most cases more proper determination is: LDS is so large collection of data that the internal storage is not able to store it. It is clear that this definition is not exact, too; following facts must be taken into account when identifying

given data set:

i) computer and its operating system,
ii) mode of coding and data representation,
iii) problem to be solved.

While the machine sets the physical limit on the maximum amount of data stored, the operating system gives the logical (and more realistic) one. Using special codings, structures and types of data representation the originally LDS may become not to be LDS one. When solving some problems several data sets are asked to be resident in the core simultaneously and this may be impossible in spite of the possibility to store each of them separately.

The most frequent types of mathematical objects represented by LDs-s are matrices (mostly the sparse ones), vectors and arrays generally. Trees, queues and other structures can also appear.

## 2.2. Components of the LDS software

There are two types of components which must be contemplated in constructing and using the LDS software:

i) program modules (software),
and ii) data files allocated outside the modules.

The program module is a separate part of a program code which communicates with other modules only by means of files. A subroutine, a program or a part of a program code can be a program module. The data file is a product of the module's activity. In contrast to the module, file is a dynamic component. LDS-s are data files, but also when it is useful to split up the module, some new "communication" data files are required to provide the connection.

It is advantageous to classify the relations between modules and files by using three classes:

i) creation a new file on the basis of given parameters or as a product of computations,
ii) use of a file as an input for the following computations,
iii) file management, i.e. copying, data structure exchanging, providing of explicitely given

modifications of some values etc.

Examples.

Counting the determinant, matrix is an input file (ii) and the result can be printed, displayed or stored into a communication file (i). The module, which solves a system of linear equations uses the matrix of the system and the vector of right sides values as the input (ii) and the resulting vector as the output (i). Exchanging the row-oriented code of given matrix to column-oriented one with possibility of keeping both files is the file managing process (iii).

## 3. LDS software developement

### 3.1. Tools and techniques

We shall point out some useful tools and techniques specific in developing the LDS software.

Since most data structures used in LDS package are fairly complicated, it is helpful to have modules which will give various information about the data in intelligible form. For example, one may ask for a zero-nonzero structure of a sparse matrix stored in packed form or for a structure of a given tree.

Other useful tools are structure debugging modules. They are used to test as much as possible whether the data contained in a storage structure is valid. To track down an error, a debugging module is inserted in strategic points of the erroneous module and this expnded program is then re-run.

Largeness of data causes some difficulties in testing. Auxiliary modules producing the test problems are desirable. The test sets chosen or generated from widely different real applications are prefered.

A lot of methods of storing the same data set are usually possible. For example, sparse matrix can be coded in many ways (see [4]). There are several manners to handle this variability:

i) construction of more modules for the same activity on different structures

ii) building modules independent on the structure

iii) using modules for exchanging the data structures as much as possible

The first possibility can rapidly increase the number of modules. For example, having three types of matrix structures, nine modules for matrix multiplication must be constructed. The second way is disadvantageous because of two main reasons: adding a new type of coding requires exchanges in more modules; the same "structure handling" program code will be present in multiple modules. So we prefere the third way. However, the preference is not strict. In some cases (e.g. solving a system of linear equations) construction of more structure dependent modules is desirable, because of the strong dependence of the complexity of some algorithms on data structures. Except of the specific tools a lot of general techniques exist. They are briefly described in [2].

### 3.2. Advantages

The great independence of modules is helpful in many ways.

The project can be divided into smaller units and the interaction between them minimized. Just this kind of division is the very best one according to [1].

Modules producing the test data as it was mentioned before enable to test the programs separately. This can rapidly save the time: when a chain of modules is constructed, anything can be tested without need of debugging of all previous modules.

The variety of programming languages may be used in constructing such a software without problems in different handling of parametrs - only the data representation must be in accord.

## 4. Adaptability and portability

### 4.1. Extension and contraction

The construction of an application software system conforming to all user's need is out of the question. Some modules are often inapplicable and some of them are frequently absent to satisfy the user's desire.

The contraction is the first stage of software adaption in which user picks out all suitable modules. The portion must be compact or it must consist of compact parts. This requirement can be easily fulfilled because of its great modularity.

Usually, the user wants to add some own modules or adapt the original ones to suit the system according to his object. When the addition suffices, only the data structures and representations are of interest to the user. The extension can be more simple if the debugging modules and other tools mentioned before are used.

The problem of extension and contraction is not specific only for applicated software, but for the software generally (see [3]).

### 4.2. Implementation

Implementation is a really serious problem in the LDS software. A lot of modules claim much time and large storage to run, therefore it is necessary to adapt them precisely to given computer. Also operating system dependence is great. These facts cause the implementation is more laborious than usually.

## 5. References

[1] Dennis, J.B.: The design and construction of software systems, in "Software Engineering" (F. L. Bauer ed.), pp. 12-28, Springer-Verlag Berlin 1977 (the first edition 1973)

[2] Gentleman, W. H., George A.: Sparse matrix software, in "Sparse Matrix Computations" (J. R. Bunch ed.), pp. 243-262, Academic Press, New York 1976

[3] Parnas, D.L.: Designing software for ease of extension and contraction, IEEE Transaction on Software Engineering, Vol. SE. 5, No. 2, 1979, pp. 128-137

[4] Tewarson, R. P.: Sparse Matrices, Academic Press, New York 1973

Verfasser:

Dr. Ľubor Kollár
Ústav aplikovanej matematiky a výpočtovej techniky
Univerzita Komenského
Mlynská dolina
842 15 Bratislava
ČSSR

Mikulecký, Peter - Kelemen, Jozef

# Supervising Expert Systems:

# An Attempt to Organize Certain Mathematical Software Intelligently

## Abstract

The paper is devoted to some aspects of the design and organization of mathematical software. An analysis of the possibility of using some recent results and ideas developed in the area of Artificial Intelligence (especially in the connection with the representation of knowledge) is sketched, and a new type of expert systems -- supervising expert systems -- is suggested. These are based on the ideas of the frame representation of knowledge, developed by Minsky, and implemented in FRL (Frame Representing Language). The final part of the paper is devoted to some considerations on exploitations of the Minsky's ideas concerning so called K-theory of memory in the design of "intelligent" software systems for solving problems in applied mathematics.

*Mephistopheles:*

...

*Wer will was Lebendigs erkennen und beschreiben,*
*Sucht erst den Geist herauszutreiben,*
*Dann hat er die Teile in seiner Hand,*
*Fehlt leider ! nur das geistige Band.*

...

J.W.Goethe Faust, I, Studierzimmer

## 1. Introduction

Recently, a new programming technology has been developed concerning the problem of transferring human expertise in given domains into effective machine form. A lot of so-called *expert systems* have been implemented, enabling computer systems to perform convincingly as advisory consultants. According to the characterisation given in [10], expert systems are program systems that are able to solve or to advise how to solve some substantial problems generally conceded as being difficult and requiring expertise. Expert systems are called *knowledge based* because their performance depends critically on the use of facts and heuristics used by experts.

Expert systems are seemed to be very useful tool for relati-

vely unskilled specialists or non-specialists in given domains. Moreover, they have been used as a vehicle for Artificial Intelligence (AI) research in the direction of a very effective exploitation of the recent results on knowledge representation and the design of programming languages for AI. A deep characterization of expert systems can be found in [5].

In the connection with the expert systems development we have turned our attention mainly on two ways of exploitation the AI research results

(1) The tools established in the scope of the AI research for the representation of knowledge (e.g., the FRL programming language, see [9]) can be successfully used for unambiguously interpretable representation of *nonalgorithmic* knowledge. This can be illustrated by representation of knowledge in medical diagnostics.

(2) The tools mentioned in (1) can be (perhaps after slight enlargement) used as tools of an "intelligent" *integration* of previously disconnected programming products. This way of using the AI research results is in our opinion very perspective and deserves our main attention all through the paper.

## 2. Supervising Expert Systems: A Motivation and Basic Considerations

A human expert is able to *integrate* in some "intelligent" way his partial abilities or knowledge into certain greater units. He is able not only to use the learned knowledge but also to establish (relatively reliable) the circumstances under which what knowledge is useful and in what sense. The knowledge enabling it is usually called *the meta-knowledge* (see, e.g. [1]). It is acceptable to state such approximation of the term "meta-knowledge" which explaines it as a knowledge on the knowledge-base having in the system disposal.

Without giving some extra reasons we can state that essentially all real "artificial" expert systems dispose with (at least implicitly included) system of meta-knowledge. For instance, Barr in [1] pointed out that it is possible to view an interpreting prog-

ram in terms of knowledge about its data base, called meta-knowledge, because the interpreter must in some sense know the extent and limit of available knowledge, what facts are relevant in a given situation, how dependable they are, and how they are used.

The "classical"style of the design of expert systems is such, that the knowledge-base and the representation of necessary meta-knowledge are established in a mutual interaction (though, usually using different means). It is possible to adjust meta-knowledge to the specific features of the knowledge-base, and vice versa. This is very often exploited and it probably causes that to the time present it is very rarely a success appeared in separating some universal normatives for the expert systems design from the experiences with designing particular expert systems.

We shall to emphasize the role of the meta-knowledge in expert systems by introducing the notion of the *supervising expert systems*. These are based on the fact that recently (particularly in some areas of science, e.g., in the applied mathematics) there are accumulated a lot of mutually *disconnected* software products (e.g., whole libraries of application programs for solution a variety of problems in applied mathematics). The knowledge of an user - expert in using such program packages is primarily not identical with the knowledge expressed in the programs of the package, but its contents mainly includes the effective way of the exploitation of the package. It means, that in the relation to the contents of a program package the expert plays the role of an intelligent *supervisor* deciding when who what will perform. The knowledge used by an expert in such activity has in the relation to the knowledge embedded in the programs of a package character of the meta-knowledge. The first goal to be achieved in the way towards artificial supervising expert systems is the transfer of such knowledge into artificial systems (computers).

Let us shortly summarize the essence of the supervising expert systems

- to profit from the existing program packages without knowing in detail the structure of such package or without necessity to construct new elements of the package;
- to enable to a non-specialist using of program packages in rather intelligent and qualified way in solving problems from

given domains;

- to facilitate to a relatively unskilled specialist using of program packages with the goal to improve his skillness in solving problems from given domains.

In what follows we try to present some arguments on behalf the thesis that existing means of AI are applicable in achieving this goal.

## 3. Supervising Expert Systems : A Case Study

In the previous section we have characterised in short what we understand under the supervising expert systems. For the sake of illustration we shall mention now several aspects of the nature of supervising expert systems. To be more concrete, we shall demonstrate an example, how can be constructed a supervising expert system for the domain of applied mathematics, and what are the differences between supervising experts and experts in the common-sense meaning.

Let us imagine the situation of a mathematician who is given a problem, say, to solve a boundary value problem for an elliptic partial differential equation. What has he to do ?

First, the mathematician must to identify the *type* of the problem in the sense of its mathematical characterization. He is exploiting his knowledge on the domain resulting into the realization, that he is solving a boundary value problem for given elliptic PDE. Then the mathematician takes into account what *requirements* have to be put on the solution of the problem, and again according to his knowledge he is choosing *a method* for solving the given problem. It is obvious, that more skilled expert in the domain has to his disposition a larger variety of methods, while a layman hardly knows about one or two suitable methods. However, one feature can be shared by both of them they essentially need not know a lot of details about used methods, or about their implementations. An analogy with the knowledge of programming languages is on hand a user creating a program need not know how are the used instructions implemented, how are they realized in the computer in detail. But an implementer, or designer of the language must have a deep knowledge in this direction.

Above mentioned feature can be considered as one of the main features of the supervising expert -ystems they are able to perform considerations and other activities like very skilled experts in the problem domain, who essentially know

- what type of problem is to be solved,
- what method is the best (according to the present circumstances) for this type of problems,
- how are the principles of using the method,
- whether or not the method is available,

but they *need not know* how is such and such method detaily constructed or implemented. That is, they can use the method as a "black box".

Let us continue our considerations on the activity of an expert - mathematician, who is given the problem of solving a boundary-value problem for an elliptic PDE. Now, he already knows the type of the problem, the requirements putting on the solution,and what methods are available on his computer. According to his skillness, he is inspecting his own knowledge or he is consulting some other specialists what method is the best for his problem, and suddenly, he is choosing the method and uses it. In the view of our considerations from the previous chapter, the mathematician is *supervising* (more or less intelligently) when who and what will do And this is actually what we require the supervising expert systems will do

Hence, a supervising expert system extends in a rather intelligent manner the abilities of an existing package of methods necessary for solving the problems in a given problem domain. It is playing a role of a skilled expert in this domain, who is supervising the employment of the methods from the program package.

Following just mentioned basic ideas, a supervising expert system has been designed at the Comenius University in Bratislava (see [6], [11]) which would be able

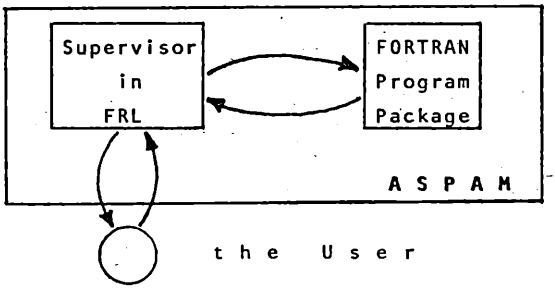
- to analyze and recognize given problems from the area of applied mathematics (more concrete, the domain is: initial and boundary value problems for ordinary and partial differential equations);
- to make an expertise in suggesting the best suitable (and, of course, available) method for solving the problem;

- to choose the method from a (rather large) program package;
- to arrange the application of the method, i.e., the computation;
- to evaluate the results and to correct eventual mistakes.

Some basic ideas of the possibility of the construction of such system have been introduced in [3].

The supervising expert system is called ASPAM (The Advisor by Solving Problems from Applied Mathematics) and for the time being has not been implemented as a whole. A part of the system has been experimentally implemented in the FRL programming language at ES-1010 computer as an illustration of the usability of the FRL language for such purposes.Further implementations are in the preparation at the ES-1033 and SM 4/20 computers.

The system ASPAM can be very schematically visualised as follows




The majority of above mentioned activities of the system are performed *interactively* in tight contact with the user. The user, according to his skillness enables to perform system's activities in more or less leeway. There is a lot of types of differential equations represented in the supervising part of the system using their frame representation in the FRL, and the system matches the input against them and recognize the problem to be solved. As an example, let us write down the FRL representation of an ordinary differential equation, say, the first order linear ODE with the constant coefficients and with a right-hand side

```
(*(DEFRAME *LDE-RHS-CC-1
           *(AKO ($VALUE (LDE-1)))
           *(RH-SIDE ($VALUE (CONST)))
           *(NEXT-MEMB ($VALUE (CONSTxY))
                       ($VALUE (ZERO)))))
```

The frame LDE-RHS-CC-1 is connected with the frame LDE-1 (which is higher in the hierarchy of used frames) using the relation AKO (*a kind of*). The frame LDE-1 represents all first order linear ODE's. In the slots RH-SIDE and NEXT-MEMB are then the values of the right hand side of the equation, and the next member of the left-hand side (the member without the derivation of y), respectively.

According to the modular construction of the ASPAM system and the modularity of the FRL programming language, the expert system can be very easily enlarged by adding some other types of equations or initial- and boundary-value problems.

## 4. Another Approach: Viewing Supervising Expert Systems As Collections of Communicating Agents

In the last years in several professional circles of the AI-researchers an opinion has appeared according to which the way of the functional and structural approximation of the intelligence can be led through the understanding of *intelligence as a performance of a collection of certain intercommunicating entities*.

Hewitt [2] presents, for example, an approach to modelling intelligence in the terms of communicating knowledge-based experts, whereby each expert can be viewed as a society that can be further decomposed until the primitive *actors* of the system are reached.

Yonezawa and Hewitt [12] state that the tools for representing the modular knowledge with procedural attachments (for example, Minsky's frames) can be modelled and implemented as actors.

While the approach just mentioned streams from the use of a new type of the *computational paradigm*, Minsky comes to the concept of the distributivity using a new *psychological hypothesis*. In the sense of this hypothesis (see, e.g., Minsky [8]) the mind is viewed as an organized society of *intercommunicating agents*. The character

of this intercommunication is sketched in [7]. Primary Minsky's intention is to establish such (computationaly formalizable) theory of the structure and function of the memory, in which besides "classical" questions of the type *"How is the information represented?"* or *"How is it stored, retrieved, etc.* interesting for those who are working in the area of AI, would be possible also effectively answer to more psychologically motivated questions of the type *"How are the abstractions made ?", "How are they later instantiated ?",* and so on.

Minsky tries to envelop all these questions by establishing the thesis, that the function of a memory is to re-create the state of the mind. The schema how this can be realized is formulated relatively simply

*"When you 'get an idea', or 'solve a problem', or have a 'memorable experience', you create something we shall call a K-lines for it.*

*This K-line gets connected to those 'mental agencies' that were recently active -- i.e., which were involved in the memorable mental event.*

*When the K-line is later 'activated', it reactivates those mental agencies, creating a 'partial mental state' resembling the original."* (Minsky [7])

It is obvious that this schema is only a starting formulation and requires further concretization before its implementation. Nevertheless, this fact cannot hinder us in considering the potential possibility of an exploitation of the schema in the design of supervising expert systems. Our considerations can be, of course, only contemplative and their goal is not in giving instructions to an activity, but only a proposition for further contemplations.

Let us suppose now, that we intent to create again a supervising expert system for applied mathematics. Every program from a program package or a library can be considered as an agent. These agents (let us denote them $K_1$, ..., $K_n$ and call them K-agents) are then able to solve some particular problems which can appear in the area of applied mathematics (e.g., a procedure for the numerical quadrature is an agent, a procedure for the Runge-Kutta method of integration of an ODE can also be viewed as an agent, etc.). We need also agents which are able to find out the relevance of some

$K_l$ to the given problem. Denote them $P_1, \ldots, P_m$ and call them the P-agents. They express certain meta-knowledge we have treated in the Section 2.

Another agents are necessary for finding out the acceptability of the results received by $K_1, \ldots, K_n$. Denote them $K^1, \ldots, K^r$. These are also K-agents.

Finally, the formulation of concrete goals of our expert system is made using a special type of agents $G_1, \ldots, G_k$ -- the G-agents.

Assume, that connections between agents are inserted into the expert system by human experts. These connections express primarily their meta-knowledge concerning the program-base over which the supervising expert system is constructed. There are several *types* of connections between agents. We shall call the connections between P-agents *P-lines*, between K-agents and P-agents *K-P-lines*, and so on.

Let us illustrate now the previous considerations. Let us suppose we are designing a supervising expert system for solving the initial-value problems for ODE's of the first order, and in the program-base we have just two well-known numerical methods -- the Runge-Kutta method of the 2nd order, and the Hamming predictor-corrector method of the 4th order. It is rather poor package, indeed, but for the sake of illustration it is enough. Assume, that the system dispose with the following agents (in the parentheses the verbal specifications of functions of the agents are expressed):

$P_1$ (in the given problem finds out the greatest derivation in the ODE)

$P_2$ (finds out the existence and the form of the right-hand side of the ODE)

$P_3$ (finds out the given interval)

$P_4$ (finds out the given integration step)

$P_5$ (finds out the form of the initial condition)

$P_6$ (establishes the required precision of results)

$P_7$ (establishes the required precision of the corrector in the Hamming's method)

$P_8$ (using the results of the activity of subordinated agents $P_1, \ldots, P_7$ establishes whether or not it is reasonable to solve given problem using the Runge-Kutta method)

$P_9$ (analogically establishes whether it is reasonable to use the Hamming's method)

$P_{10}$ (using messages from all subordinated P-agents establishes whether the given problem is to solve an ODE, i.e., whether or not the obtained data are relevant to the formulated problem)

The problem to be solved can be formulated using unique G-agent $G_1$ which arranges connections between the data about the problem and the data about the result (the solution of the problem), and enables to specify the type of the problem.

K-agents enabling the proper process of solution the given problem can be established as follows

$K_1$ (the procedure for the Runge-Kutta method of the 2nd order, which uses on its input the results of the perceptions of the agents $P_1$, ..., $P_6$ mediated through $P_8$)

$K^2$ (tests the agreement of the result with required precision, as a result of the perception of $P_6$)

$K_3$ (the procedure for the Hamming's predictor-corrector method of the 4th order, which needs in addition the result of the perception of $P_9$)

$K_4$ (arranges the output of obtained results)

The connections between agents are schematically visualised on Fig.1 by arrows. The type of each connection can be derived from the type of its tail-agent and head-agent. When the connection begins, for example, in the P-agent and ends in the K-agent, the type of this connection is P-K. It is worth to note, that we assume, that given agent can treat an information and to pass it to another agent only in the case all its inputs (of the same type) are saturated. The numbers assigned to the connections are the numbers of stages of the transfer of the information between agents (note, that the possibility of the parallelism can be considered).

In our considerations on the Minsky's suggestion we have not dealt with the characterization of the information retrieving mode in the nets of agents. Note, that the mode of the information retrieval in our example is in accord with the Minsky's suggestion (recall, e.g., the clockwise spiral form of computation, stressed in [7]).

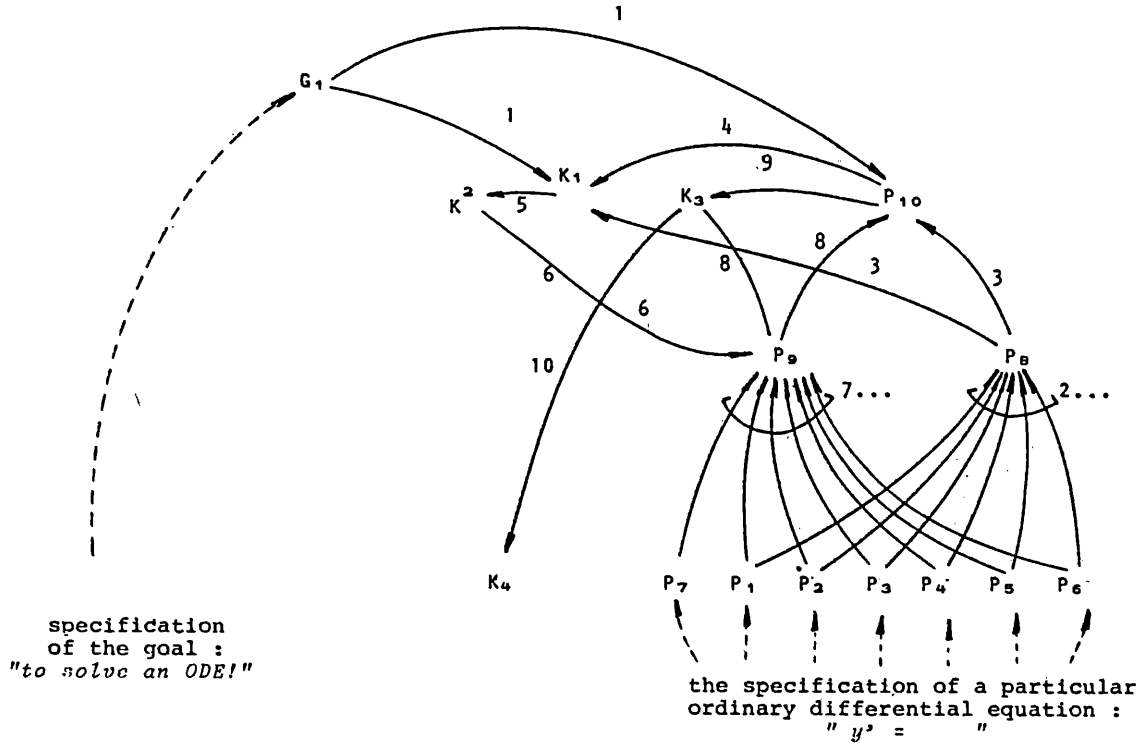

1
G1
1
4
9
K1
K2
5
K3
P10
8
8
3
3
6
6
10
P9
P8
7...
2...
K4
P7
P1
P2
P3
P4
P5
P6
specification
of the goal :
"to solve an ODE!"
the specification of a particular
ordinary differential equation :
" y' = "

Fig. 1

## 5. Conclusion

Our intention in the paper was to characterize one relatively new type of expert systems -- the supervising expert system -- mainly by introducing two suggestions of concrete supervising expert systems construction for the purposes of solving problems in the area of applied mathematics.

We suppose, that a powerful tool for the realization of expert systems can be found among programming languages designed in the scope of Artificial Intelligence. We have already some first experience with using FRL for this purpose (compare [11], [6]). Note, that there exist already in the area of actor's approach some suitable programming tools, for instance the languages PLASMA [2] or Act 1 [4]. However, we still do not know about their using in the process of design of the expert systems. Alike we have no knowledge on the existence of a computer realization of the Minsky's model.

Nevertheless, we can suppose that just by the pragmatism influenced considerations on potential practical exploitation of these suggestions can done some motivations to the creation of a more precise image of the requirements for concrete computer implementations.

## References


[1] Barr,A.: *Meta-knowledge and Memory*. Heuristic Programming Project, Working Paper HPP-77-37, Stanford Univ., 1977

[2] Hewitt,C.: Control Structure as Patterns of Passing Messages. In: *Artificial Intelligence: An MIT Perspective, Vol.2* (Eds. Winston,P.H., Brown,R.H.), MIT Press, Cambridge, Mass., 1979, pp. 435-465

[3] Kelemen,J. - Mikulecký,P.: Levels of Proceduralization in Applied Mathematics. In: *Proc.Nat.Conf. "ALGORITMY'81"*, Vysoké Tatry, April 1981 (In Slovak)

[4] Lieberman,H.: *A Preview of Act 1*, MIT AI Memo No.625, MIT, Cambridge, Mass., June 1981

[5] Michie,D.: Expert Systems. *The Computer Journal 23*,1980,pp. 369-376

[6] Mikulecký,P. - Štolcová,E.: *ASPAM - An Example of Supervising Expert System*. (In preparation)

[7] Minsky,M.: K-Lines: A Theory of Memory. *Cognit.Sci.4*,1980, pp. 117-133

[8] Minsky,M.: The Society Theory of Thinking. In: *Artificial Intelligence: An MIT Perspective, Vol.1* (Eds. Winston,P.H., Brown,R.H.), MIT Press, Cambridge, Mass., 1979,pp.423-450

[9] Roberts,R.B. - Goldstein,I.P.: *The FRL Manual*. MIT AI Memo No.409, MIT, Cambridge, Mass., 1977

[10] Stefik,M. et al.: The Organization of Expert Systems - A Tutorial. *Artificial Intelligence 18*,1982, pp. 135-173

[11] Štolcová,E.: *Design of an Expert System for Solving Problems in Applied Mathematics*. Diploma Thesis, Dept. of Theor.Cybernetics, Comenius Univ., Bratislava, 1982,(In Slovak)

[12] Yonezawa,A. - Hewitt,C.: Modelling Distributed Systems. In: *Machine Intelligence, Vol.9* (Eds. Hayes,J.E., Michie,D., Mikulich,L.V.), Ellis Horwood, Chichester, 1979, pp. 41-50

Verfassers

*RNDr. Peter Mikulecký, CSc.*
*Ústav aplikovanej matematiky a*
*výpočtovej techniky,*
*Univerzita Komenského,*
*Mlynská dolina*
*842 15 Bratislava*
*ČSSR*

*RNDr. Jozef Kelemen*
*Katedra teoretickej kybernetiky,*
*Matematicko-fyzikálna fakulta,*
*Univerzita Komenského,*
*Mlynská dolina*
*842 15 Bratislava*
*ČSSR*

## Konzeption eines Auswertungssystems für das Krankenhaus Friedrichshain von SKR-Rechentechnik

Durch den immer größer werdenden Umfang der Nutzung von Datenverarbeitungstechnik im Krankenhaus ergeben sich weiterführende Fragen und Anforderungen. So steht inzwischen ein umfangreicher Datenpool zur Verfügung, für den der Zugriff verhältnismäßig schnell und zu jedem Zeitpunkt möglich ist. Es ergibt sich die Frage nach Auswertung dieser vorhandenen Daten. Eine häufige Fragestellung betrifft die Anwendung statistischer Verfahren. Deshalb wird für das Krankenhaus Friedrichshain ein Auswertungssystem entwickelt, mit dem diese Fragen beantwortet werden können.

### 1. Allgemeine Voraussetzungen

Um hier sinnvolle Grenzen zwischen Möglichem und sinnvoll Durchführbarem zu finden, ist es notwendig, sich einerseits das Spektrum der häufig auftretenden Fragen anzusehen und andererseits die Voraussetzungen der künftigen Nutzer zu beachten. Das Spektrum der Fragestellungen ist, wie in einem allgemeinen Krankenhaus zu erwarten, relativ breit gefächert. Um eine sinnvolle Größe des Auswertungssystems zu erhalten, muß hier zwischen häufig gebrauchten und sehr speziellen am Rande auftretenden Fragestellungen unterschieden werden. Als vernünftiger Rahmen erscheint hier die Beschränkung auf häufiger benötigte Verfahren, wobei jederzeit der Anschluß spezieller Verfahren möglich ist.
Da relativ einfache Fragestellungen oft auftreten, soll der Anwender seine Probleme über ein speziell dafür bereitgestelltes Terminal soweit wie möglich selbst lösen können. Da EDV-Kenntnisse nicht vorausgesetzt werden dürfen, muß dem Nutzer weitestgehend Hilfestellung gegeben werden. Das läßt sich am besten über Dialogsteuerung verwirklichen.
So ergeben sich für das Auswertungssystem folgende Rahmenbedingungen:

- es ist für die Anwendung in einem größeren Krankenhaus gedacht, um mit Hilfe eigener Kleinrechentechnik schnelle Auswertungen an Ort und Stelle zu ermöglichen.

- Anwender, die häufigere Fragestellungen in dieser Richtung haben, sollen mit dem System an einem dafür vorgesehenen Terminal nach kurzer Einführungszeit selbständig arbeiten können.
- Der Umfang der angebotenen Verfahren soll überschaubar und für normalerweise häufig auftretende Probleme zugeschnitten sein. (Es soll also bei weitem nicht der Leistungsumfang von z.B. PP-Statistik erreicht werden, zumal es sich hier in diesem Fall um den Einsatz von Kleinrechentechnik handelt, dies soll und kann nicht Ziel für diese Anwendung sein.)
- Der Nutzer soll soweit durch Dialog unterstützt werden, daß möglichst keine EDV-Kenntnisse notwendig sind, um Fragestellungen selbst bearbeiten zu können.
- Das System muß variabel sein, um eine große Anpassung an das Aufgabenprofil zu erreichen.

2. Konzeption des Auswertungssystems

Die Konzeption des Auswertungssystems sieht 3 Teilbereiche vor:

- Datenbereitstellung
- Transformation von Daten
- die eigentlichen statistischen Verfahren.

Datenbereitstellung

Zum Bereich der Datenbereitstellung gehört die Möglichkeit, Datenfelder einzugeben (wahlweise über verschiedene periphere Geräte), sie zu ändern, Daten einzufügen, Tabellen der Daten auszugeben, Datenfelder zu überprüfen (auf bestimmte Randbedingungen, um Fehler weitestgehend auszuschalten) und zu löschen.

Der Nutzer kann hier zwischen den einzelnen Verfahren wählen und kann jederzeit einen gewählten Arbeitsgang abbrechen, um mit einem anderen fortzufahren.

Nachdem die Daten in der gewünschten Form zur Verfügung stehen, erweisen sich oftmals Transformationen als zweckmäßig.

Transformation von Daten

Hier soll dem Nutzer die Möglichkeit gegeben werden, über die Verknüpfung vorhandener Operatoren (+,•,-.: Fkt) die gewünschten Transformationen anzugeben und auf bestimmten Datenfeldern anzuwenden.

eigentliche statistische Verfahren

Jetzt sind die Datenfelder in der Form vorhanden, wie sie für die Anwendung eines statistischen Verfahrens benötigt werden. Anschließend wird das entsprechende Verfahren aufgerufen. Es ist möglich, von einem Verfahrensbereich zu einem anderen zu springen, je nach Bedarf ist so eine relativ optimale Bearbeitung einzelner Fragestellungen möglich.

3. Konzeption der Dialoggestaltung

Um eine konkrete Aufgabe zu lösen, sind jeweils nur bestimmte Berechnungen und Verfahren notwendig. So ist es erforderlich, den Rechenprozeß entsprechend steuern zu können. Weiterhin ist es wichtig, dem Anwender einfach und übersichtlich die Auswahl von Verfahren bereitzustellen und ihm durch einfache, möglichst an keine bestimmte Formen gebundene, Eingabemöglichkeiten die Arbeit zu erleichtern.
Somit ergeben sich für die Dialoggestaltung zwei wichtige Aufgaben:

- die Steuerung des Rechenprozesses selbst (Ablaufsteuerung)
- Unterstützung bei der eigentlichen Lösung, wobei hier folgende fälle zu unterscheiden sind:
  Formulierung der gewünschten Eingabeparameter mit entsprechender Fehlerkontrolle und, falls notwendig, wiederholtem Aufruf zur Eingabe
  - Meldung von Fehlern und Hinweise zu ihrer Beseitigung
  - Angebot von Verfahrenslisten
    Hinweise auf Voraussetzungen, die bestimmte Verfahren verlangen, und Angebot entsprechender Prüfverfahren (so daß grobe Anwendungsfehler vermieden werden können)
    Soweit möglich Angebot der Ergebnisse in verbaler Formulierung.

Diese Hilfen werden bereitgestellt, um den Rechenprozeß für einen Laien handhabbar zu gestalten und ihn damit die selbständige Lösung zu ermöglichen. Der Dialog kann und soll aber nicht Kenntnisse der medizinischen Statistik ersetzen, er kann aber bei vorhandenen Kenntnissen oder entsprechender vorheriger Information Hilfestellung im Umgang mit den Verfahren bieten.

Die genannten Anforderungen sind mit Hilfe der SKR-Rechner SM4-10, SM4-20 gut realisierbar. Durch Benutzung von Overlay-Strukturen und die Steuerung über Indirekt-File-Kommandos ist der zur Verfügung stehende HS-Platz optimal ausnutzbar. Ebenso wird von Datenmengen ausgegangen, die auch von Kleinrechentechnik sinnvoll bewältigt werden können; für extreme Anwendungsfälle ist das Auswertungssystem nicht ausgelegt.

Verfasser:

Dipl.-Math. Renate Schirrow
Städt. Krankenhaus im Friedrichshain
Organisations- und Rechenzentrum
1017 Berlin
Leninalle 49

Haberland, Heinrich
Roscher, Günther

# Erfahrungen bei der Gestaltung und mehrfachen Nutzung vorgefertigter Grundsoftware in der technischen Vorbereitung

## 1. Einführung

Die technische Vorbereitung der Produktion beinhaltet das Vorausdenken von Erzeugnissen und ihrer Herstellung. Die Rationalisierung dieses informationsverarbeitenden Prozesses erfordert den Einsatz der Rechentechnik. Unter der rechnergestützten technischen Vorbereitung (CAE) verstehen wir das Erarbeiten, Speichern und Ausgeben der technischen Dokumentation durch die EDVA auf der Grundlage der Kommunikation zwischen Ingenieur und Rechner. Schwerpunkte der Nutzung der Leistungsfähigkeit der EDVA zur Unterstützung der Konstrukteure und Technologen sind:

- die Speicherung der Daten und Programme,
- die Zugriffe auf die Daten,
- die Durchführung von Berechnungen und Entscheidungen,
- die Ausgabe der technischen Dokumentation.

Mit den EDVA des ESER ist eine einheitliche, leistungsfähige Rechentechnik eingesetzt, die über einfache und programmierbare Abonnentenpunkte vom Arbeitsplatz der Ingenieure aus genutzt werden kann. Dabei steht die Entwicklung von Rationalisierungslösungen für die technische Vorbereitung als Herausforderung an die schöpferische Tätigkeit der Ingenieure und Wissenschaftler. Sie erfordert die Systematisierung und Abstraktion der wissenschaftlich-technischen Erkenntnisse und Erfahrungen von Konstrukteuren und Technologen.

Das Ziel ist dann die Umsetzung der abstrahierten Ergebnisse als Modelle in der Form von Daten und Programmen. Dadurch wird es möglich, diesen durch hohe Kompliziertheit und Komplexität gekennzeichneten informationsverarbeitenden Prozeß teilweise oder ganz dem Rechner zu übertragen.

## 2. Strategie und Entwicklung vorgefertigter Grundsoftware für die rechnergestützte technische Vorbereitung

Die für ESER bereitgestellten Betriebssysteme sind durch hohe Leistungsfähigkeit gekennzeichnet. Sie erfordern für ihre Nutzung die Spezialkenntnis der Rechentechnik.
Rechnergestützte Lösungen zur technischen Vorbereitung müssen jedoch bevorzugt von Konstrukteuren und Technologen mit hoher Fachkenntnis erarbeitet werden. Die Komplexität und Kompliziertheit der zu erarbeitenden Modelle erfordert die volle Ausschöpfung der Leistungsfähigkeit moderner EDVA. Deshalb ist es notwendig, daß durch Spezialisten der Softwareentwicklung verallgemeinerte rechnerorientierte Komponenten als Grundsoftware bereitgestellt werden, die die Konstrukteure und Technologen in die Lage versetzen, ohne Spezialkenntnisse der Rechentechnik ihre Lösungen problemorientiert zu erarbeiten und anzuwenden. Die Umsetzung dieser Strategie führt zur Entwicklung von 2-Komponenten-CAD-software, wie sie von Krause /1/ beschrieben wurde.
Diese verallgemeinerten, rechnerorientierten Komponenten tragen den Charakter eines technischen Datenbankbetriebssystems (DBBS), das von kommerziellen DBBS signifikant verschieden ist. Mit Hilfe dieses DBBS können die Konstrukteure und Technologen den Rechner unmittelbar zur Realisierung ihrer betriebsspezifischen Modelle nutzen.
Die vom Datenverarbeitungszentrum Magdeburg im Auftrag des VEB Werkzeugmaschinenkombinat "7. Oktober" Berlin entwickelte Rationalisierungslösung RTV /2/ genügt den definierten Anforderungen von 2-Komponenten-CAD-software. RTV besteht aus der rechentechnischen Grundsatzlösung, mit deren Hilfe eine Reihe von Anwenderlösungen gestaltet wurden und noch gestaltet werden. (Bild 1).
Aus den Erfahrungen und Ergebnissen dieser nach einheitlichen Methoden entwickelten Anwenderlösungen resultiert die technologische Grundsatzlösung, die verallgemeinerte Modelle der Anwenderlösungen beinhaltet.

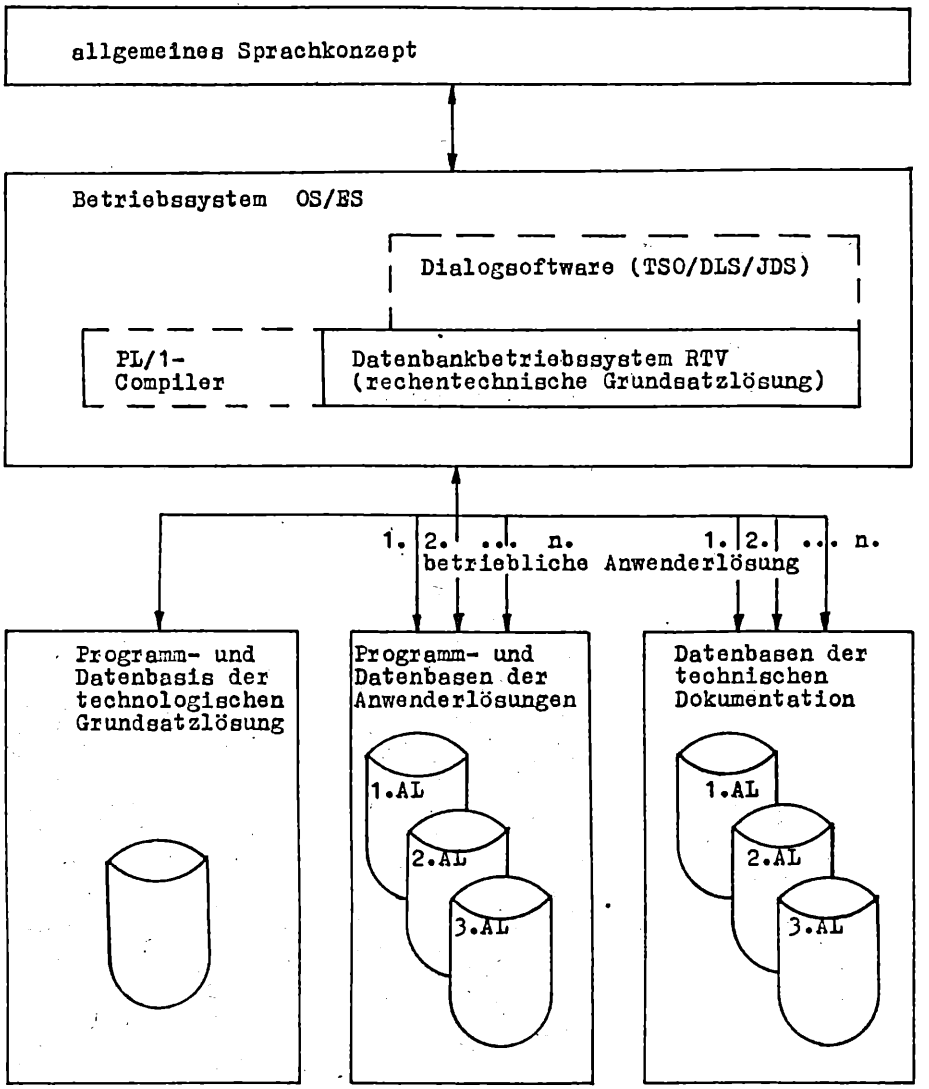



Bild 1: Übersicht zur Rationalisierungslösung RTV

## 3. Die rechentechnische Grundsatzlösung RTV – ein technisch orientiertes Datenbankbetriebssystem

### 3.1. Sprachkonzept

In internationalen Veröffentlichungen wird als Programmiersprache für die rechnergestützte technische Vorbereitung in starkem Maße auf FORTRAN orientiert. Das hat seine Begründung unter anderem in der historischen Entwicklung und in der Forderung nach Portabilität der Software. Es gibt jedoch bereits neuere Systementwicklungen, die als entscheidenden Vorteil gegenüber entsprechenden FORTRAN-Implementierungen moderne Programmiersprachen nutzen und die dadurch nach eigener Aussage einen Qualitätssprung erreichen konnten /3/. Die erhöhte Leistungsfähigkeit der modernen Programmiersprachen drückt sich u. a. aus in:

- flexibale Verwendung von Zeichen- und Bitketten, Strukturen und Listen,
- interaktive Anwendung,
- Einbindung von Programmen, die in anderen Sprachen programmiert sind (FORTRAN, ASSEMBLER),
- klare Strukturierung.

Da im RGW mit dem ESER einheitliche Rechentechnik genutzt wird, ist hier die Portabilität hinsichtlich der eingesetzten Großrechner gelöst.
Ausgehend von diesen Gesichtspunkten wurde PL/1 als Datenmanipulationssprache für RTV genutzt. Die Anwendung der Programmiersprache PL/1 und der in sie eingebetteten Zugriffstechniken des DBBS ist in der methodischen Richtlinie zur Programmierung von Anwendersprachen beschrieben. In Anlehnung an moderne Programmiersprachen wurde ein allgemeines Sprachkonzept (Bild 2) entwickelt, das auf die Aufgabenklasse der deskriptiven Sprachen für die technische Vorbereitung ausgedehnt ist.
Die Schlüsselworte (Wortsymbole) der deskriptiven Sprachen und der Aufbau der Programmsätze und Programme sind durch den Anwender entsprechend der ausgewählten Aufgabenklasse frei wählbar. Sie werden auf der Grundlage des allgemeinen Sprachkonzeptes vereinbart.

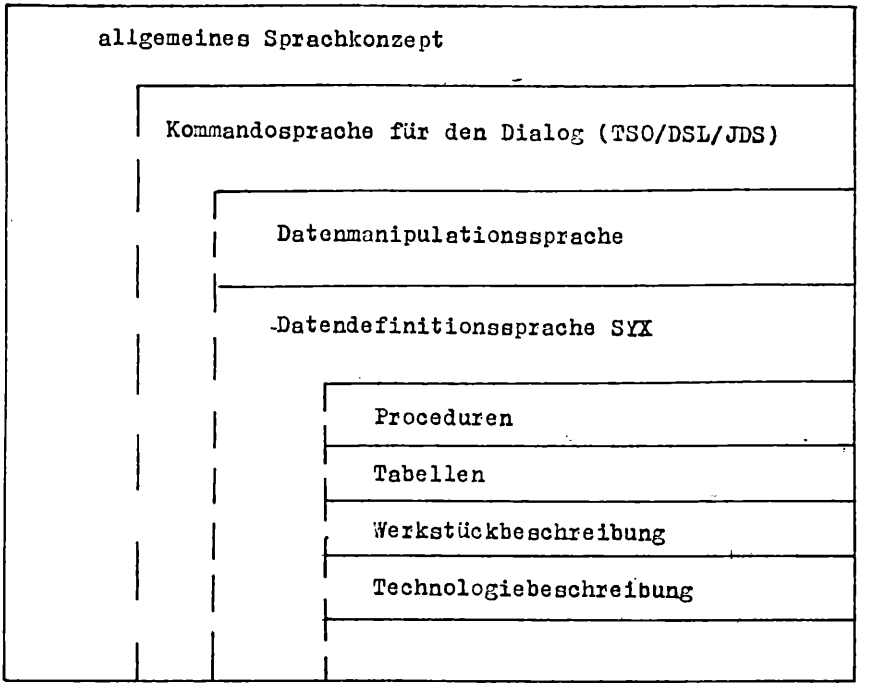



Bild 2: Sprachkonzept für die rechnergestützte technische Vorbereitung

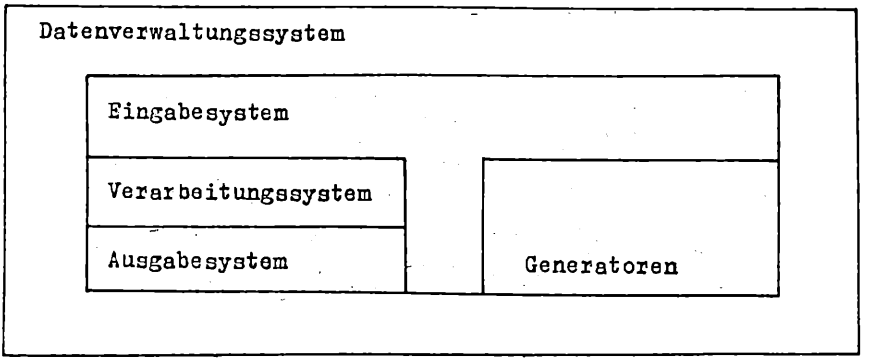



Bild 3: Datenbankbetriebssystem für die rechnergestützte technische Vorbereitung

Eine methodische Richtlinie zur Vereinbarung und Verwaltung von Eingabesprachen einschließlich der Datendefinitionssprache SYX unterstützen den Anwender bei der Entwicklung seiner Sprachen.

Für folgende Aufgaben wurden bisher Sprachen vereinbart:

- Werkstückbeschreibung,
- Technologiebeschreibung,
- Arbeitsgangausarbeitung,
- maschinelle NC-Programmierung,
- Tabellen zur Speicherung von Normativen, Matrizen und Katalogen,
- Proceduren zur Speicherung beliebig strukturierter Daten,
- festformatige Sprachen zur Belegerfassung,
- Recherche in den Daten der technischen Dokumentation,
- Moduldefinition,
- Datendefinition.

Mit Hilfe der methodischen Richtlinie und der Sprache SYX sind die Anwender in der Lage, selbständig Sprachen zu vereinbaren und an die notwendigen Veränderungen anzupassen.

## 3.2. Datenverwaltung

Alle Eingabedaten liegen unabhängig von der Sprache in der Form von Quellprogrammen vor und werden mit den Komponenten des Betriebssystems verwaltet. Vorteilhaft wirkt sich hierbei eine leistungsfähige EDIT-Funktion eines angeschlossenen Dialogsystems aus. Die realisierte Möglichkeit des dynamischen Aufrufes von Problemprogrammen der Anwender erschließt besonders günstige Möglichkeiten. Die Problemprogramme können dadurch unabhängig voneinander geändert, übersetzt und gespeichert werden. Somit ist ein einfacher Änderungsdienst und eine unproblematische Mehrfachnutzung möglich.
Aufbauend auf diesen Erfahrungen wurde ein Modularkonzept entwikkelt und der Systementwicklung von RTV zugrunde gelegt /4/.
Alle Moduln und Daten werden über ihren Namen aufgerufen. Die Verbindungen zwischen den Moduln werden nur zum Zeitpunkt des Aufrufes geknüpft.
Der Zugriff von den Problemprogrammen auf die Daten der Anwender wird durch einfache und erweiterte Zugriffstechniken unterstützt.

Diese Zugriffstechniken werden über den dynamischen Unterprogrammaufruf aktiviert. Somit kann über das DBBS (Bild 3) auf Tabellen, Proceduren, Werkstückbeschreibung, Technologiebeschreibung und die Daten der erstellten technischen Dokumentation zugegriffen werden. Beim Zugriff auf Tabellen können gleichzeitig mehrstufige Entscheidungskriterien zur Auswahl der Daten wirksam werden.
Die vom Anwender eingespeicherten Tabellen und Proceduren werden durch das DBBS in eine speicher- und abarbeitungsgerechte Form transformiert.
Besondere Aufmerksamkeit wurde der Speicherung und der Auswertung der Werkstückbeschreibung gewidmet. Sie wird in eine interne Normalform transformiert, die die Struktur und die Daten der Werkstückbeschreibung beinhaltet.
Die einfachen Zugriffstechniken gestatten zum Beispiel:

- die Bereitstellung eines beliebigen Maßes am Werkstück,
- die Festlegung einer Zusatzangabe zu diesem Maß,
- die Anfrage, ob ein Werkstück ein bestimmtes Formelement besitzt mit gegebenenfalls gleichzeitiger Bereitstellung der Parameter des Formelementes.

Die erweiterten Zugriffstechniken beinhalten unter anderem den Komplexzugriff auf alle Daten der Werkstückbeschreibung einer definierten Werkstückklasse oder einer Struktur innerhalb einer Werkstückbeschreibung. Dadurch wird besonders die Entwicklung von Anwenderlösungen unterstützt, die das Variantenprinzip nutzen.
Analog werden die Zugriffe auf die einfachen Datenstrukturen der Technologiebeschreibung und der übrigen Eingabedaten gelöst.
Spezielle Komponenten des DBBS sichern die Verwaltung und den Änderungsdienst der technischen Dokumentation, speziell der Arbeitspläne und der Strukturlisten.
Wird vom Anwender ein eigenes kommerzielles Datenverwaltungssystem für diese Probleme eingesetzt, so werden die Anschlußbedingungen gesichert.

## 3.3. Generatoren zur Anpassung der Grundsatzlösung

Für definierte Teile der Anwenderlösung wurden spezielle Generatoren erarbeitet, die aus den mit Hilfe der Datendefinitionssprache SYX durch den Anwender erstellten Beschreibungen spezifische Moduln erzeugen.

Das sind besonders:

- anwenderabhängige Teile des Sprachübersetzers für deskriptive Sprachen,
- Moduln zur Anpassung der Dateiverwaltung und der Zugriffstechniken,
- anwenderspezifische Moduln zum Belegdruck, zur Ausgabe der Steuerinformation für NC-Maschinen und für Zeichentische sowie zur Speicherung der technischen Dokumentation.

Die Anleitung zur Generierung wird in der methodischen Richtlinie zur organisatorischen Anpassung der Anwenderlösungen gegeben.

## 4. Erfahrungen bei der mehrfachen Anwendung

Seit 1977 wird die Rationalisierungslösung RTV im Routinebetrieb auf ESER-Anlagen genutzt /5/. Mit der Realisierung weiterer Anwenderlösungen wurde die Leistungsfähigkeit, Allgemeingültigkeit und Anwenderfreundlichkeit der Lösungen schrittweise erhöht. Anhand einer methodischen Richtlinie zur betrieblichen Anpassung und Entwicklung sind die Konstrukteure und Technologen der Anwenderbetriebe in der Lage, selbständig ihre Anwenderlösungen zu gestalten und weiterzuentwickeln. Durch zentral organisierte und durch betriebsbezogene Schulungen wurden ausgewählte Kader (Systemingenieure) speziell für diese Aufgaben qualifiziert.
In jedem Anwenderbetrieb wurde eine Arbeits- und Forschungsgemeinschaft gebildet, in der die späteren Nutzer der Rationalisierungslösung, die Systemingenieure, betriebliche Organisatoren und die Entwickler der rechentechnischen Grundsatzlösung gemeinsam an der Realisierung dieser Aufgaben arbeiten. Diese Zusammenarbeit ist gekennzeichnet durch Praxisnähe, eine straffe, vertraglich verankerte Organisation und determinierter Arbeitsteilung.

| Programm- und Datenbasis der technologischen Grundsatzlösung |
| --- |
| (allgemeingültige Anwendermoduln)<br>• Problemprogramme (Berechnungen, Entscheidungen)<br>• Tabellen )Normative, Kataloge, Standards,...)<br>• Proceduren (Datenstrukturen, Texte,...)<br>• SYX-Standardsprachvereinbarungen |

| Programm- und Datenbasis der Anwenderlösung |
| --- |
| • Problemprogramme<br>• Tabellen<br>• Proceduren<br>• SYX-Sprachvereinbarungen<br>• Werkstückbeschreibungen<br>• Technologiebeschreibungen |

| Datenbasis der technischen Dokumentation |
| --- |
| • Stücklisten<br>• Arbeitsplanstammkarten |

Bild 4: Programm- und Datenbasen

Sie hat eine entscheidende Rückkopplung auf die Weiterentwicklung der rechentechnischen Grundsatzlösung.

Die einheitliche Organisation der Erarbeitung von Anwenderlösungen, die einheitlichen methodischen Richtlinien und die einheitliche rechentechnische Grundsatzlösung bewirken, daß die erarbeiteten anwenderspezifischen Moduln zwischen den Anwendern ausgetauscht werden können. Im Rahmen einer überbetrieblichen Nutzergemeinschaft wird zielgerichtet daran gearbeitet, allgemeingültige, nachnutzungsfähige Moduln zu entwickeln oder als Vertragsforschung entwickeln zu lassen. Diese Moduln stehen in der zentralen Programm- und Datenbasis (Bild 4), der technologischen Grundsatzlösung, allen Nutzern zur Verfügung.

Die technologische Grundsatzlösung beinhaltet:

- Problemprogramme,
- Tabellen und Proceduren,
- SYX-Standardsprachvereinbarungen.

Bei der Verwaltung der technologischen Grundsatzlösung gelten folgende Grundsätze:

Der Erarbeiter eines Moduls übernimmt dessen langfristige Pflege und Wartung. Bei inhaltlicher Erweiterung des Moduls muß dieser unter einem neuen Namen eingespeichert werden. Zur Beschreibung und Verwaltung der Moduln wird eine Moduldokumentationssprache bereitgestellt.

## 5. Schlußbemerkungen

Die Entwicklung der Rationalisierungslösung RTV als allgemeine verfügbare Grundsoftware mit daraus abgeleiteten betrieblich genutzten Anwenderlösungen hat sich bewährt /6/, /7/. Besonders vorteilhaft ist dabei die unabhängige Entwicklung, Bereitstellung und Wartung der rechentechnischen Grundsatzlösung für eine Vielzahl von Anwendungsfällen in Einheit mit der selbständigen Gestaltung und schritthaltenden Adaption von Anwenderlösungen durch betriebliche Entwicklerkollektive. Ein wichtiges Merkmal ist die Herausarbeitung verallgemeinerter Moduln der Anwenderlösungen und ihre Einbindung in eine technologische Grundsatzlösung für alle Nutzer.

Dadurch wird der Entwicklungsaufwand für neue Anwendungsfälle weiter reduziert bei gleichzeitiger Erhöhung ihres Niveaus und ihrer volkswirtschaftlichen Wirksamkeit.

## 6. Literaturhinweise

/1/ Krause, F.-L.
Methoden zur Gestaltung von CAD-Systemen.
Dissertation, Berlin 1976

/2/ Becher, H.; Haberland, H.; Mayer, K.; Roscher, G.
RTV - Eine Rationalisierungslösung zur rechnergestützten technologischen Vorbereitung.
COMCONTROL '79, Konferenzmaterial Nov. 1979, Sopron

/3/ Schlechtendal, Bechler, Enderle, Leinemann, Olbricht
CAD-Berichte / REGENT-Handbuch.
Kernforschungszentrum Karlsruhe 1978

/4/ Haberland, H.; Mayer, K.; Roscher, G.
Einsatz der Rationalisierungslösung RTV - rechnergestützte technologische Vorbereitung - zur Gestaltung einer technologischen Grundsatzlösung.
INTERTECHNO '81, Konferenzmaterial Nov. 1981, Budapest

/5/ Entwicklerkollektiv RTV
RTV-Rechnergestützte technologische Vorbereitung.
VEB DVZ Magdeburg, VEB WMK "7. Oktober" Berlin
Broschüre 1978

/6/ Haberland, H.; Kautz, K.; Roscher, G.; Zerressen, H.
RTV-rationalisiert die technologische Vorbereitung.
rechentechnik/datenverarbeitung, Heft 7, 1981

/7/ Strempel, H.; Roscher, G.; Haberland, H.
Automatisierte Systeme der Informationsverarbeitung in der technologischen Vorbereitung der Produktion.
5. Lehrbrief
Automatisierung der technologischen Produktionsvorbereitung.
Zentralstelle für das Hochschulfernstudium,
Dresden, Sep. 1981

Verfasser:

Dr.-Ing. Heinrich Haberland
VEB Datenverarbeitungszentrum Magdeburg
3018 Magdeburg, Lübecker Str. 2

Dr.-Ing. Günther Roscher
VEB Werkzeugmaschinenkombinat "7. Oktober" Berlin
VEB WMW Ingenieurbetrieb Berlin
1120 Berlin, Liebermann Str. 27/37

Aden, Horst

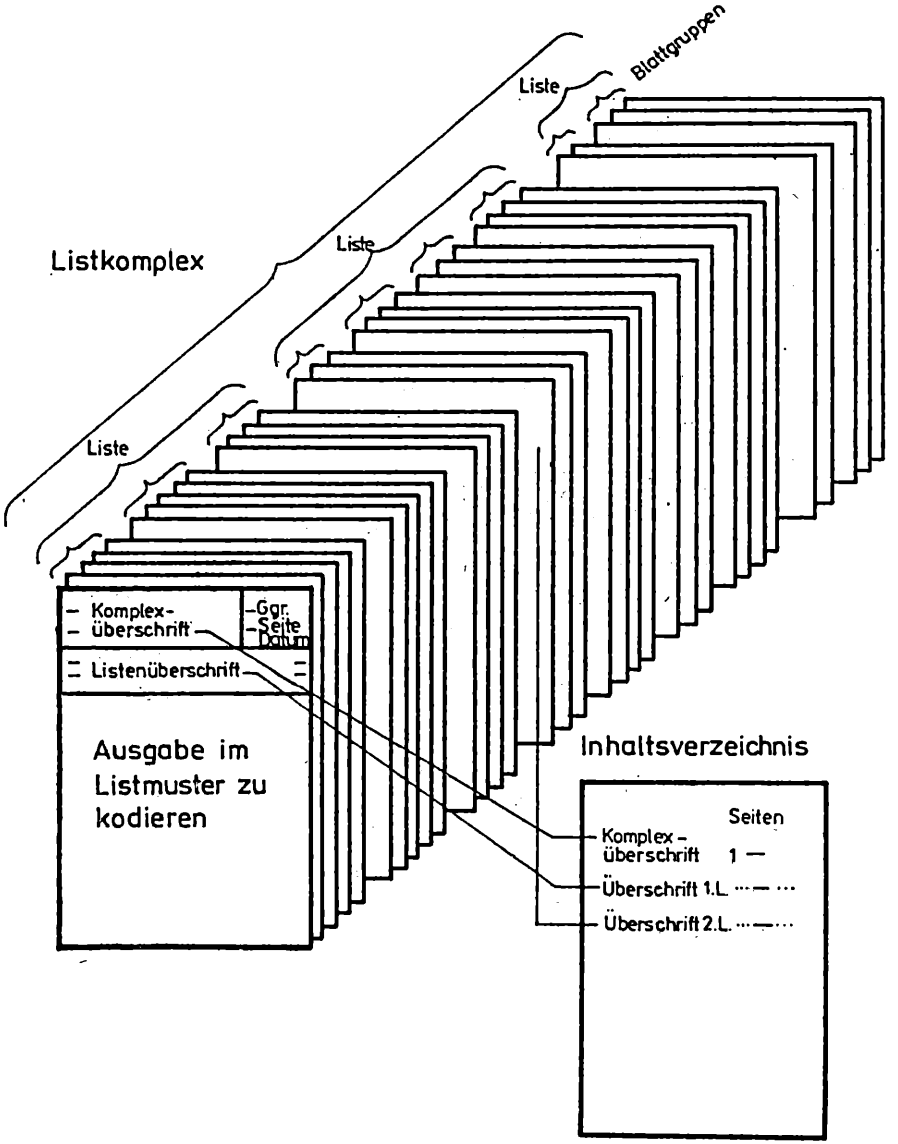
## Listgenerator LSDAT

Der Listgenerator LSDAT dient der Auswertung sequentieller Datenbestände und liefert in einem Arbeitsgang Zusammenstellungen von verschiedenartigen Listen zu geschlossenen Dokumenten (Listkomplexen, siehe Abb. 1).
Zur Kodierung von LSDAT-Anweisungen sind keine Programmierkenntnisse erforderlich. Vorausgesetzt wird nur, daß der Anwender mit dem Aufbau von Datensätzen mit Ordnungsbegriffen, den Prinzipien ihrer Sortierung und der Gruppenbildung vertraut ist.
LSDAT ist eine Komponente des Datenverarbeitungssystems VIV2/BOM [1] und somit von der Pflege der Anweisungen bis zur Verwaltung der Datenbestände technologisch unterstützt. Die Generierungstechnologie des VIV2/BOM gestattet dem Anwender, mit formalen Speichernamen zu arbeiten, ohne sich um die konkrete Lokalisierung der aktuellen Speicher kümmern zu müssen.
Alle Anweisungen sind in der vom Assembler und den ES/OS-Dienstprogrammen her bekannten Form zu kodieren, wobei die Syntax der Schlüsselwortoperanden problembedingt erweitert worden ist. Schlüsselwörter können soweit verkürzt werden, wie ihre Eindeutigkeit zuläßt. Damit wird ein Minimum an Kodieraufwand bei hoher Flexibilität und Erweiterungsfähigkeit der Sprache erreicht.
Die Anwendung von LSDAT setzt einen vom Anwender seinen Bedürfnissen entsprechend einzurichtenden DASS-Referenzspeicher voraus. Dieser Speicher bedarf keiner Reorganisation und kann wahlweise mit Sicherung der Umsätze zur automatischen Rekonstruktion betrieben werden. Die Pflege der gespeicherten Daten ist an das Programmsystem PORG [2] angelehnt und damit sehr einfach.

1) Eine Sondernummer der Zeitschrift "Rechentechnik - Datenverarbeitung" zu VIV2/BOM wird vorbereitet.

2) Das Programmverwaltungssystem PORG wird vom Kombinat Robotron vertrieben.

Abb. 1 Schematische Darstellung eines Listkomplexes mit Inhaltsverzeichnis



Alle Datenstrukturen werden mit Datenbeschreibungstafeln festgelegt. Jedes Datenelement dieser Strukturen wird mit einem Namen versehen und durch eine Relativadresse, seine Länge und einen Datentyp charakterisiert. Der Datentyp hat eine funktionelle Komponente, die angibt, ob das Datenelement eine Ordnungsfunktion (Identifikator) hat und eine Komponente, die die Darstellungsform beschreibt.
Darstellungsformen sind: Zeichenketten, Bittketten, Zahlen.
Datenbeschreibungstafeln sind als Objekte des Referenzspeichers zu speichern.
Jedem im Datensatz angegebenen Ordnungsbegriff muß im Referenzspeicher eine Nomenklatur zugeordnet werden. Die Nomenklatursätze enthalten den Schlüssel und weitere Informationen wie Texte, Sortiernummern, Angaben zur Maßeinheitenbehandlung und anderes nach Belieben des Anwenders. Die Struktur der Nomenklatursätze wird mit Datenbeschreibungstafeln festgelegt. Die Nomenklaturen werden den Ordnungsbegriffen mittels Begriffstabellen zugeordnet. Auf dieser Basis können Listmuster entworfen, kodiert und in den Referenzspeicher eingegeben werden. Nach jeder Eingabe oder Pflege eines Listmusters tritt der Listmusterkompiler in Aktion und erzeugt ausführbare Programmkode und Datenbereiche, die im Referenzspeicher als weitere Segmente zum Listmusterquelltext gespeichert werden.
Ein Listmuster wird mit einer Identifikationsanweisung (LMID) eingeleitet, in der die Datenbeschreibungstafel zu den Datensätzen genannt wird. Über die Ordnungsbegriffe der Datensätze wird damit auch der Zugang zu den Nomenklatursätzen des Referenzspeichers gewährleistet. Mit Anweisungen zur Vereinbarung von Variablen (LMVP) werden Arbeits- und Parameterfelder festgelegt und wahlweise initialisiert und bei Bedarf ihr zulässiger Wertevorrat festgelegt.
Durch die Vereinbarung synthetischer Ordnungsbegriffe (LMVS) und ihre Versorgung mit Werten mittels Bedingungen an den Datensatz insbesondere an die Ordnungsbegriffe, gelingt es, Datensätze zusammenzuführen, die sonst mit keinerlei Sortierung nach den Ordnungsbegriffen zusammengeführt werden können.

In einer Anweisung zur Ordnung der Daten (LMOG) werden die Ordnungsbegriffe in der Reihenfolge ihres Ranges aufgeführt und mit dem Hinweis: Sortierung nach den Schlüsseln oder den Ordnungsnummern aus der zugehörigen Nomenklatur versehen. Durch Einfügen von Schaltervariablen besteht die Möglichkeit, die Gestaltung der Listen zu bereichern.
Letzten Endes werden die Daten mittels einer Reihe von Listmustersatzanweisungen (LMSA), die über ihren Ordnungsbegriffs- oder Schalternamen den Stufen der Listmusterordnungsanweisung zugeordnet sind, ausgegeben. Die Listmustersatzanweisungen sind entweder vom Typ Kopf und werden zum Beginn der Gruppe entsprechend dem Gruppenwechsel oder einer Schalterstellung nach fallender Hierarchie ausgeführt oder sie sind vom Typ Fuß und werden am Ende der Gruppe vom niedrigsten Niveau beginnend bis zur höchsten Gruppenwechselstufe oder dem höchsten Schalter in der Stellung "ein" ausgeführt. Vor jeder Ausgabe wird, falls vorhanden, eine zugehörige Aktion ausgeführt. Die Aktionen bestehen aus Folgen von Ergibtanweisungen bzw. bedingten Anweisungen. In den Aktionen können Rechnungen ausgeführt werden. Die logischen Ausdrücke für die Bedingungen und die arithmetischen Ausdrücke für die Rechnungen werden in der aus der Elementarmathematik bekannten Form kodiert. Mittels sogenannter Listfunktionen ist es möglich, insbesondere kompliziertere logische Bedingungen leicht zu formulieren.
In einem Arbeitsgang können Daten in mehreren Listen mit unterschiedlicher Sortierung entsprechend den vorbereiteten Listmustern ausgegeben werden. Zu diesem Zwecke ist ein Abruf zu kodieren. Der Abruf steuert die Ausgabe eines oder mehrerer Listkomplexe. Jede Ausgabe von Listkomplexen wird durch eine LK-Anweisung eingeleitet, die im wesentlichen die Listkomplexüberschrift und Angaben zur Geheimhaltung enthält. Danach folgen LM-Anweisungen, die das jeweilige zu benutzende Listmuster und die Listenüberschriften festlegen. Auf jede LM-Anweisung folgen Anweisungen, die der Versorgung des Listmusters mit Parametern und Daten dienen. In einer Gruppe von Ordnungsbegriffen können mehrere verschiedene Listen ausgegeben werden, indem in der Folge der LM-Anweisungen zu den unterzuordnenden Listen ein entsprechender Parameter kodiert wird.

Die Listung wird mittels einer Steueranweisung gestartet, in der die Eingabespeicher, die Datenstruktur, der Abruf, die Ausgabespeicher und technologischen Parameter zu den Arbeitsspeichern angegeben werden. Die auf magnetische Datenträger ausgegeben Listkomplexe können mittels DIAL am Bildschirm betrachtet und mittels MBD in beliebiger Anzahl und auch auszugsweise gedruckt werden.
Mit dem folgenden Beispiel wird die Kodierung einer Datenbeschreibungstafel, eines Listmusters, eines Abrufs und einer technologischen Anweisung angedeutet.

## Beispielliste

DOKUMENT NR. D GEH 3782/5
SEITE: 1
DATUM: 10.05.1982

KAPITEL NR. K
YYY

OB1: ob1 ob1-text
OB2: ob2 ob2-text

| OB3 / OB4 | | SPALTEN-UEB 1 | SPALTEN-UEB2 | SPALTEN-UEB 3 |
|---|---|---|---|---|
| ob3 | ob3-text | | | |
| ob4 | ob4-text | df1 | df2 | df1-df2 |
| ob4 | ob4-text | df1 | df2 | df1-df2 |
| | ⋮ | | | |
| *** | | s1 | s2 | s1-s2 |
| ob3 | ob3-text | | | |
| ob4 | ob4-text | df1 | df2 | df1-df2 |
| ob4 | ob4-text | df1 | df2 | df1-df2 |
| | ⋮ | | | |

Darin bedeuten:

OB1,OB2,OB3,OB4 - willkürliche Namen für Ordnungsbegriffe mit den Schlüsseln ob1,ob2,ob3 und ob4

ob1-text, usw. - Texte zur Erläuterung der Bedeutung der Schlüssel ob1 usw., die in den Feldern TEXT der Nomenklatursätze zu OB1 usw. enthalten sind. Adressierung OB1.TEXT usw.

df1,df2 - Werte aus den Feldern DF(1) und DF(2) der Datensätze

s1,s2 - Summen der Werte df1,df2 innerhalb der Gruppen zu OB3

Die Datenbeschreibungstafel DSDBT für die Datensätze lautet:

```
BSR 6Ø              Datensatzlänge
...
STR OB1,O,N,2,1     Ordnungsbegriffe (O)
STR OB2,O,N,2,3     numerisch linksbündig
STR OB3,O,N,4,5     gepackt (N) in 2 bzw. 4 Bytes,
STR OB4,O,N,2,9     d.h. vier- oder achtstellig
   ...              an den Adressen 1,3,5 bzw. 9
STR DF,D,E,4,13,12  Datenfeldvektor (D) mit zwölf Feldern vom
                    Gleitkommatyp (E) der Länge vier ab Adresse
                    13 im Datensatz
```

Der Einfachheithalber wird für alle Nomenklaturen die folgende gleiche DBT zugrunde gelegt:

```
BSR 41              Nomenklatursatzlänge
SOR OB3             Sortierung im Referenzspeicher nach den
                    Schlüsseln
STR OB1,O,N,2,1
STR OB2,O,N,2,1     Schlüsselvereinbarungen,
STR OB3,O,N,4,1     alle Schlüssel ab Adresse 1
STR OB4,O,N,1,1
STR LFDN,D,H,2,5    laufende Nummer für Sortierung in den
                    Listen
STR TEXT,D,T,21,7   Text zum Schlüssel
STR AWN,D,B,13,28   Bitvektor zur Bildung von Teilnomenklaturen
```

Kodierung des Listmusters für die Beispielliste

```
Name des Listmusters
LMID NAME=LMBSP,DBT=DSDBT
Sortierung und Satzfolge festlegen:
LMOG (NAME=OB1,ONR= WDS),        SORT NACH SCHLUESSEL                  *
     (NAME=OB2,AWS=(1020,7049,3001,4081)),  AUSWAHL UND                *
                                            SORTIERSCHLUESSEL
     (NAME=OB3,ONR=LFDN),        SORT NACH LFDN DER NOMENKL           *
     (NAME=OB4,ONR=SORTVAR)      SORT IM ABRUF FESTLEGEN
Variable festlegen:
LMVP (NAME=SORTVAR,TYP=DC,LNG=8,INIT='LFDN',                          *
      WVRT=('LFDN',' WDS')),                                          *
     (NAME=S,TYP=DD,ANZ=2        SUMMENREGISTER
Anweisungen zur Berechnung und Ausgabe der Daten:
Die Anweisung mit NAME= SWB wird bei jedem Blattwechsel durch
Wechsel der Schlüssel OB1 oder OB2 und bei jedem Blattüberlauf
ausgeführt. WZ (Zeilenwechsel), RS (Lücke), A und F sind Format
elemente.
LMSA NAME=OB1,TYP=KO,FORMAT=(,WB)      BLATTWECHSEL
LMSA NAME=OB2,TYP=KO,FORMAT=(,WB)      BLATTWECHSEL
LMSA NAME=#SWB,TYP=KO,FORMAT=(%SWBFMT)      SPALTEN
LMSA NAME=OB3,TYP=KO,AKTION=(S(1)=0,S(2)=0),
     FORMAT=((OB3,OB3.TEXT),(WZ,A,RS(6),A))
LMSA NAME=OB3,TYP=KO,                                                 *
     AKTION=(S(1)=S(1)+DF(1),S(2)=S(2)+DF(2)),                        *
     FORMAT=((OB4,OB4.TEXT,DF(1),DF(2),DF(2)-DF(1)),                  *
        (WZ,RS(4),A,RS(2),A,RS(1),(3)F(9,2)))
LMSA NAME=OB3,TYP=FU,FORMAT=(('****',S(1),S(2),S(2)-S(1)),            *
   (WZ,(A(4),RS(n),(3)F(9,1)))
```

Anweisung zur Ausgabe der Spaltenüberschrift, auf die mit %SWBFMT bezug genommen wird

```
SWBFMT FORMAT ,WB,('OB:',OB1,OB1.TEXT),(A,A,RS(2),A)          ✱
              ('OB2:',OB2,OB2.TEXT),(RZ(1),A,A,RS(2),A),      ✱
              (n)'-',(RZ(1),A(n)),                            ✱
              'OB3              ',(RP(2,1),A),                ✱
                  OB4           ',(RP(3,1),A),                ✱
              ' SPALTEN- SPALTEN- SPALTEN-  ',(RP(2,n),A),    ✱
                UEB1     UEB2     UEB3       ,(RP(3,n),A),    ✱
              (n)'-',(RZ(1),A)
```

Kodierung des Abrufs

Listkomplex einleiten

```
LK FORM=62,EXPL=(5,2),GGR=GEH,GNR=3782/5,                     ✱
   UEB11='DOKUMENT NR. D',UEB21=' ',UEB31='                   ✱
   SNR=4
```

Listmuster auswählen und mit Überschrift versehen

```
LM NAME=LMBSP,BSPR=1,UEB11='KAPITEL NR. K',                   ✱
   UEB21=YYY,IVZ1=...
```

Wertzuweisung an Variable im Listmuster

```
LV SORTVAR= WDS
```

Datenauswahl für das Listmuster. Es sind zwei Teilmengen auszuwählen. In der ersten Teilmenge sollen nur Sätze ausgewählt werden, in denen OB1 Schlüssel enthält, deren zugeordnete Nomenklatursätze im Feld AWN in den Positionen 1,5,7 oder 20 eine 1 enthalten. Der Schlüssel OB2 kann beliebige Werte annehmen, wird aber durch LMOG auf die dort angegebenen vier Schlüssel beschränkt. Der Schlüssel OB3 kann nur ganze Hunderterwerte annehmen. OB4 darf nur ungleich 0000 sein. Die zweite Teilmenge wird neben den Bedingungen an die Ordnungsbegriffe durch eine Bedingung an die Datenfelder bestimmt.

```
                          ↓ — entspricht &
   OB1.AWN:(1,5,7,20),OB2:✱✱✱✱,                              ✱
   OB3:(✱✱ØØ),OB4 =0000
AW OB1.AWN: 6,OB2:✱✱✱✱,OB3:(0000-5000,9000),                  ✱
:  OB4.LFDN:(200-700),,(DF(2)-DF(1)/DF(1) 100
LM
LK
:
```

Die technologische Anweisung lautet:

```
./ LSDAT DBT=DSDBT,ABRUF=ABRB,LISTEN=LST1,                    *
./ INHALT=SYSA,ARB=?AUSW,                                     *
./ EINO1=STAMMN
```

Aden, H., Berlin

**V i e r k e** , Bärbel
**H a a g e n** , Erhard

Entscheidungstabellenvorübersetzer "WET"

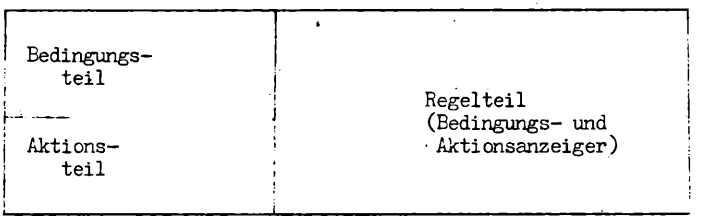
## 1. Grundsätze

Entscheidungstabellen gewinnen in der Praxis immer größere Bedeutung. Sie ermöglichen dem Fachpersonal, ihre Probleme in einem schnell erlernbaren, EDV-gerechten System zu formulieren.
Allerdings zeigt es sich, daß der EDV-gerechte Entwurf in Entscheidungstabellen und ihre direkte Umsetzung in eine Programmiersprache den Anforderungen der Praxis nicht gerecht wird. Verlangt wurden, neben der einfachen und gemischten Notation, weitgehende Prüfungen und Optimierungen. Im Gegensatz dazu stand die Forderung nach Geschwindigkeit an zweiter Stelle. Wir haben uns deshalb entschlossen, alle verfügbaren Tests zum Ermitteln von widersprüchlichen und redundanten Regeln aufzunehmen. Die Listung aller Entscheidungstabellen während der Umformung und Optimierung und die Aufnahme einer Laufzeitunterstützung zur Ermittlung von Bedingungskonstellationen, die nicht in der ursprünglich entworfenen Tabelle enthalten sind. Für die Implementation wurde der Markierungsalgorithmus verwendet. Gegenüber dem Verzweigungsalgorithmus ist er zwar leichter zu implementieren, dafür ist er aber langsamer. Dieser Nachteil wird aber durch die umfangreichen Tests ausgeglichen.

## 2. Der Entscheidungstabellenvorübersetzer "WET"

WET ist ein Entscheidungstabellenvorübersetzer für die Programmiersprache PL/1 unter der Steuerung des OS/ES (MVT/MFT).

### 2.1. ET-Aufbau




Folgende Notationen sind dabei möglich:

#### 2.1.1. Begrenzte ET

| | | |
|---|---|---|
| Bedingungszeile: | IF A=B | Y N - ..... Y N |
| Aktionszeile: | A=B*C | X - X ..... - X |
| oder: | PERFORM AA | - X X ..... X X X |

Aktionen und Bedingungen müssen jeweils geschlossen im Bedingungs- bzw. Aktionsteil notiert werden. Es handelt sich dabei um gewöhnliche PL/1-Statments ohne Semikolon. Reicht der Platz im Aktionsteil nicht aus, so kann die PERFORM Anweisung benutzt werden.
Trifft für eine Regel die angegebene Bedingung zu, so ist ein "Y" anzugeben. Trifft sie nicht zu, ist ein "N" zu notieren. Ist die Aussage der Bedingung ohne

Bedeutung, so daß ein "-" (DON T CARE) zu schreiben. Für auszuführende Aktionen wird in der Regel ein "X" geschrieben. Für nicht auszuführende Aktionen ein "-". Sollen zur Laufzeit Aktionen ausgeführt werden, für die keine Bedingungsanzeiger in der ET enthalten sind, so sind diese Aktionsanzeiger als letzte Spalte in der ET ohne Bedingungsanzeiger zu notieren. Dies stellt die ELSE-Regel dar.

2.1.2. Erweiterte ET

```
Bedingungszeile:   IF A        =B   B       B
Aktionszeile:      A=          B*C          C
oder:              PERFORM     AA   AB
```

Die erweiterte Notation ermöglicht es, in einer Zeile mehrere Bedingungen bzw. Aktionen unterzubringen. Dies fördert die Übersichtlichkeit und Lesbarkeit der Tabellen. Treffen Bedingungen oder Aktionen für eine bestimmte Regel nicht zu, so sind an dieser Stelle Leerzeichen zu schreiben.

2.1.3. Gemischte Tabelle

Es ist möglich, die beiden zuerst genannten Formen in beliebiger Abfolge der Zeilen gemeinsam zu notieren.

## 2.2. Leistungsparameter

Zur Beschreibung jeder ET werden folgende Parameter verlangt:

NAME = Name der ET (genau 5 Zeichen)

SIZE = (Bedingungsanzahl, Aktionsanzahl, Regelanzahl)

Es gelten folgende Begrenzungen:

Bedingungsanzahl + Aktionsanzahl = 100

Regelanzahl = 100

TEXTSIZE = (Breite des Bedingungs-, Aktionsteils, Regelbreite)

Es gelten folgende Begrenzungen:

Breite des Bedingungs-, Aktionsteils = 72

Standard, wenn ausgelassen = 48

Regelbreite = 70

Standard, wenn ausgelassen = 1

BITSIZE = Maximum über die Anzahl aller Bedingungen bzw. Aktionen der im Programm auftretenden Entscheidungstabellen.

Dieser Wert wird zur Generierung des PL/1-Textes benötigt.
Die Begrenzungen im SIZE-Parameter sind Konstanten und lassen sich abhängig vom vorhandenen Hauptspeicherplatz beliebig vergrößern.

## 2.3. Wahlfreie Parameter

SORT = (LINES I COLUMNS I ROWS)

Der Parameter ermöglicht eine Sortierung nach Zeilen (ohne jede Einschränkung möglich), oder nach Spalten oder Zeilen und Spalten (dies kann zu Fehlern führen, wenn die Regeln nicht den P-Test bestehen).

OPTIONS= (Parameterliste)

Parameter:

ELSE - muß angegeben werden, wenn die ELSE-Regel Anwendung findet.

LIST, - im erzeugten PL/1-Text wird die Original ET als Kommentar gelistet

TESTLIST - alle Stufen der ET-Umwandlung (von einer erweiterten oder gemischten in eine einfache) und Optimierung werden während des Precompilerlaufes aufgelistet

OPEN - die ET wird am Regelende verlassen, die Steuerung geht zum nächsten Statment nach der ET über

CLOSE - die ET wird nach Regelende erneut durchlaufen, die Tabelle muß durch einen Sprung im Aktionsteil verlassen werden

INTERN - aus der ET wird eine Statmentfolge generiert

EXTERN - die ET wird als PL/1-Prozedur generiert

PTEST - PTEST wird ausgeführt, fällt er negativ aus, wird das Programm abgebrochen.

Die Abfolge und Nutzung der Parameter in der Optionsliste ist wahlfrei.

## 2.4. Zusammenstellung der Tests

Folgende Tests werden von WET unbedingt durchgeführt:

### 2.4.1. Redundanz

Zwei Regeln sind identisch. Eine Regel wird gelöscht.

### 2.4.2. Widerspruch

Zwei Regeln mit gleichem Bedingungsteil haben unterschiedliche Aktionsteile, der WET-Lauf wird abgebrochen.

Durch die Option "PTEST" werden folgende Tests durchgeführt:

### 2.4.3. PTEST

Dieser Test nach Pollak garantiert, daß alle Regeln der ET beliebig vertauschbar sind. Die erfolgreiche Durchführung des Tests ist Voraussetzung für ein Spalten- oder reihenweises Sortieren der Tabelle und für alle folgenden Tests.
Fällt der PTEST negativ aus, wird der WET-Lauf abgebrochen.

### 2.4.4. 1. ET-Gesetz nach Greve

Zusammenfassung von zwei Regeln, die sich nur in einem Bedingungsanzeigerpaar (Y,N) unterscheiden.

### 2.4.5. 2. ET-Gesetz nach Greve

Zusammenfassung von zwei Regeln, die sich nur in einem Bedingungsanzeigerpaar (Y,-) oder (N,-) unterscheiden.

2.4.6. 3. ET-Gesetz nach Greve

Unterscheiden sich zwei Regeln in mehreren Bedingungsanzeigerpaaren (Y,–) und/oder (N,–), so werden sämtliche darin enthaltenen Redundanten Regeln gestrichen.
Die Anzahl der "–" Anzeiger ist für ein Regelpaar auf 7 begrenzt.

Die Optimierungen 2.4.4. bis 2.4.6. werden jeweils zyklisch wiederholt, bis kein entsprechendes Regelpaar mehr auftritt.

## 3. Schlußbemerkung

Der Übersetzer befindet sich seit einem Jahr in der Praxisanwendung für kleine und mittlere Probleme, besonders bewährt hat sich dabei die Umstellung und Optimierung der Tabellen durch den Vorübersetzer in der Entwurfsphase. Es konnten dabei bereits inkorrekte und unvollständige Aussagen entdeckt und korrigiert werden.

## Literaturhinweise

Roger u. Greve Entscheidungstabellen als systematisches Hilfsmittel in der ADV-Bürotechnik 1/72

Horst Strunz Entscheidungstabellentechnik ZO 2/73 und 3/73

Verfasser:

Dipl.-Math. Bärbel Vierke
WTZ Potsdam
Systemprogrammierung
1500 Potsdam
Helmholtzstr. 6/7 Tel.: 20321

Dipl.-Math. Erhard Haagen
Fachschule f. Gesundheits- u. Sozialwesen
ORZ
1500 Potsdam
Berliner Str. 114/115 Tel.: 4641

Böhme, Peter

## Die Anwendung des Programmpaketes SYSDFV zur Simulation spezieller, durch Differentialgleichungen beschriebene Systeme

### Vorbemerkungen

Ziel des Beitrages ist es, das im ORZ der THLM entwickelte Programmpaket SYSDFV zu charakterisieren. Diskutiert wird vor allem die Programmstruktur und es werden die Konzepte dargelegt, nach denen die Dialogfähigkeit von SYSDFV hergestellt und die Einbindung 'externer' Programmbausteine in SYSDFV realisiert werden soll. Eine ausführliche Darlegung des Konzepts von SYSDFV (einschließlich Darstellung der Numerik und Angabe eines Anwendungsfalles) wird in /1/ gegeben.

### Anwendungsbereich von SYSDFV

Mit Hilfe des Programmpaketes SYSDFV können
- Systeme von quasilinearen parabolischen Differentialgleichungen 2. Ordnung in einer Ortsvariablen,
- nichtlineare Zwei-Punkt-Randwertaufgaben

gelöst werden. (Für halblineare bzw. lineare Gleichungen parabolischen Typs stehen spezielle Algorithmen zur Verfügung) SYSDFV ermöglicht die Behandlung einfacher kombinierter Modelle, die Durchführung von Modellsimulationen wird unterstützt. Schnittstellen für eine Arbeit im interaktiven Betrieb sind vorhanden. An der Ausfüllung dieser Schnittstellen sowie an der Schaffung von Bausteinen zur Behandlung von Modellgleichungen in zwei Ortsvariablen wird gegenwärtig gearbeitet.

Das Programmpaket wurde bisher zur Simulation von chemischen Rohrreaktoren eingesetzt. Es kann zur Simulation vieler Prozesse verwendet werden, in denen Diffusion, Konvektion, Reaktion eine Rolle spielen (Wärmeleitungs- und Verbrennungsvorgänge, chemische und biochemische Reaktionen usw.).

### Kurzcharakteristik von SYSDFV

Das Spektrum der Anforderungen an Software zur numerischen Lösung von partiellen Differentialgleichungen ist sehr breit.

Dies resultiert zum einen aus der recht umfangreichen und nicht unkomplizierten Theorie und Numerik partieller Differentialgleichungen, zum anderen aber auch aus dem größer werdenden Kreis derer, die mit unterschiedlichsten Kenntnissen und mit unterschiedlichsten technischen Voraussetzungen partielle Differentialgleichungen lösen wollen bzw. müssen. Die Wünsche gehen dabei von der 'einfachen' Lösung einer Modellgleichung bis zu umfangreichen Modellsimulationen. SYSDFV versucht diesem breiten Anwendungsspektrum entgegen zu kommen. Dies wird durch eine streng modulare Struktur und durch die Schaffung verschiedener Nutzungsebenen zu erreichen versucht.

1. Nutzungsebene: Konzipiert für einen 'ungeschulten' Nutzer, der ein 'Standardproblem' lösen will.
2. Nutzungsebene: Konzipiert für einen 'etwas qualifizierteren' Nutzer, der Probleme zu lösen hat, die nicht der Standardform entsprechen (z.B. einfache kombinierte Modelle), der einen Dialog mit dem Lösungsverfahren führen will oder der spezielle Modellsimulationen durchzuführen hat.
3. Nutzungsebene: Konzipiert für einen 'Fachmann', der mit Hilfe von SYSDFV-Programmbausteinen neue numerische Verfahren implementieren will.

Den Kern von SYSDFV bilden die Steuerprogramme, die so konzipiert sind, daß sie auch als Grundgerüst für weitere Software-Entwicklungen genutzt werden können. Die Steuerprogramme können sowohl über speziell auf SYSDFV zugeschnittenen als auch über 'extern' entstandenen Programmbausteinen operieren. SYSDFV kann charakterisiert werden als ein Programmpaket, das bestimmte Eigenschaften eines Simulationssystems und einer Fachsprache besitzt und das seine qualitative und quantitative Erweiterung unterstützt.

Datenorganisation

Die Bereitstellung der Eingabedaten erfolgt in einer speziellen Eingabesprache, die Eingabeeinheit ist vom Anwender frei wählbar (Standard: Lochkartenleser). Alle Programmsteuerparameter besitzen Standardwerte, es müssen nur die Parameter eingegeben werden, deren Wert verändert werden soll (Verwen-

dung der NAMELIST-Dateien in FORTRAN IV). Die Datenausgabe erfolgt mit Hilfe spezieller Unterprogramme, die Ausgabeeinheit ist vom Anwender frei wählbar (Standard: Drucker). SYSDFV stellt Bausteine zur Ausgabe zur Verfügung, der Einsatz anwenderspezifischer Ausgabeprogramme wird unterstützt. Moduln zur Dateneingabe über ein Terminal werden gegenwärtig erarbeitet, sie lehnen sich an das oben erwähnte Konzept an. Auch eine Datenübermittlung an das Terminal wird ermöglicht.

Aufbau von SYSDFV

SYSDFV besteht aus einer Anzahl von Programmbausteinen (Subroutinen), die in einem bestimmten Verhältnis zueinander stehen. Die Übergabe der Programmsteuerung an SYSDFV erfolgt durch einen Unterprogrammaufruf. Der Anwender kann hierfür, entsprechend den Nutzungsebenen, unter verschiedenen Unterprogrammen wählen. Diese unterscheiden sich durch

- den Service, der dem Anwender geboten wird,
- die Anpassungsfähigkeit an spezielle Aufgabenstellungen,
- die benötigten Ressourcen an Speicherplatz als auch an Rechanzeit,
- den Umfang an Kenntnissen, die der Anwender besitzen muß.

Der Lösungsprozeß (einschließlich Datenein- und -ausgabe und eventueller Kommunikation mit dem Anwender) wird durch ein Steuerprogramm organisiert. Es wird entweder das Steuerprogramm selbst aufgerufen (hoher Freiheitsgrad und hoher Aufwand für den Anwender) oder ein dem Steuerprogramm übergeordnetes Programm, durch welches bestimmte Standardannahmen getroffen werden (geringerer Freiheitsgrad und geringerer Aufwand für den Anwender). Auf der höchsten Nutzungsebene steht eine automatische Speicherplatzanforderung und -verwaltung zur Verfügung.

Alle inhaltlich eigenständigen Funktionen (sowohl den Lösungsprozeß als auch die Datenkommunikation betreffend) sind in getrennten, austauschbaren Programmbausteinen angesiedelt. Dadurch wird eine hohe Variabilität und Anpassungsfähigkeit erreicht. Durch einen speziellen Programmbaustein, der (bei der Lösung parabolischer Differentialgleichungen) nach jedem Integrationsschritt aufgerufen wird, ist sowohl ein 'echter' als auch ein 'vorprogrammierter' Dialog mit dem Lösungsver-

fahren möglich. Mit seiner Hilfe kann vorgenommen werden:
- eine Modifizierung oder ein Wechsel des Lösungsverfahrens,
- eine Änderung von Programmsteuerparametern,
- eine Modifizierung der Modellgleichungen (Simulation von Steuer-, Regel- und Störvorgängen),
- die Lösung von Modellgleichungen, in denen z.B. partielle Differentialgleichungen mit gewöhnlichen Differentialgleichungen oder mit algebraischen Gleichungen gekoppelt sind,
- eine Arbeit mit adaptiven Diskretisierungsnetzen,
- ein Test auf Stationarität der Lösung usw.

Für einfache Varianten des 'vorprogrammierten' Dialogs (die vorgesehenen Entscheidungen werden in herkömmlicher Weise vom Programm getroffen, ohne das der Anwender in den laufenden Lösungsprozeß eingreifen kann) stehen Programmbausteine zur Verfügung, ebenso werden für einfache Varianten des 'echten' Dialogs (der Anwender kann in den laufenden Lösungsprozeß eingreifen) Standardbausteine bereitgestellt.
Die Mehrzahl der oben aufgeführten Entscheidungsmöglichkeiten ist jedoch zu speziell, um sinnvoll in einem Standardbaustein untergebracht werden zu können, d.h. der Dialogbaustein wird in diesen Fällen vom Anwender selbst geschaffen. Dabei wird er von SYSDFV weitgehend unterstützt. Die Führung eines 'echten' Dialogs kann in einfacher Weise in FORTRAN programmiert werden, die Kommunikation mit dem Terminal wird über Unterprogrammaufrufe realisiert. Erste Unterprogramme hierfür (in Assembler realisiert) stehen für die EDVA R 21 (mit der Bedieneinheit als Terminal) zur Verfügung. Die Schaffung von Moduln für eine Dialogarbeit im Betriebssystem OS/ES unter Regie des TSO ist vorgesehen.

Zur Ankopplung 'externer' Moduln an SYSDFV

Zur Lösung vieler Probleme existiert bereits bewährte, leistungsfähige Software. Dies trifft z.B. auch auf das Gebiet der numerischen Lösung von gewöhnlichen Differentialgleichungen zu. Welche Vor- bzw. Nachteile bringt es, wenn versucht wird, unter Rückgriff auf vorhandene Programmbausteine neue Software, z.B. zur Lösung von partiellen Differentialgleichungen zu entwickeln? Ein wesentlicher Vorteil ist in der

Möglichkeit zu sehen, die Entwicklungs- und Testzeit für das neue Softwaresystem entschieden zu verringern. Nachteilig dagegen ist, daß es im Zusammenwirken von unabhängig voneinander entstandenen Programmbausteinen unweigerlich 'Reibungsverluste' gibt: Es erhöht sich der Rechenzeit- und Speicherplatzbedarf gegenüber 'maßgeschneiderten' Systemen. Die Frage, ob nun der Vor- oder der Nachteil schwerer wiegt, kann nicht generell entschieden werden, sondern stets nur im Zusammenhang mit dem jeweiligen Anwendungsfall. Dabei sollten vor allem folgende Aspekte geprüft werden:

- Rechtfertigt der durch 'Maßschneidern' mögliche Effektivitäts- bzw. Qualitätsgewinn den erhöhten Entwicklungsaufwand ?
- Kann in der zur Verfügung stehenden Entwicklungszeit mit den vorhandenen Kapazitäten überhaupt eine höhere Qualität erreicht werden ?

'Maßschneidern' führt in vielen Fällen zu einer guten Spezialsoftware, das Einbeziehen 'externer' Moduln bzw. das Schaffen der Möglichkeit einer solchen Einbeziehung dagegen kann zu einer guten Standardsoftware führen. Bezieht man spezielle 'externe' Moduln ein, so kann eine solche Standardsoftware sogar Spezialfälle lösen. Die Praxis zeigt, daß sowohl 'maßgeschneiderte' Software als auch (gute) Standardsoftware benötigt wird.

Natürlich führt das Eingliedern bewährter Programmbausteine in ein neues Softwarevorhaben nicht automatisch zu hoher Qualität. Diese steht und fällt mit dem Konzept dieses Softwarevorhabens und mit der Art und Weise, wie die Eingliederung der entsprechenden Programmbausteine erfolgt.

SYSDFV wird an geeigneten Stellen (z.B. zur Lösung von gewöhnlichen Differentialgleichungssystemen) extern entwickelte Programmbausteine ankoppeln. Dafür gelten folgende Grundsätze:

- Das Systemkonzept wird unabhängig von den sich zur Ankopplung anbietenden Softwarebausteinen entwickelt.
- Eingriffe in die anzukoppelnden Softwarebausteine werden nicht vorgenommen.

Beide Grundsätze sind sinnvoll: Die Programmbasis kann erweitert werden, ohne das Systemkonzept ändern zu müssen , die

eingebundenen Bausteine bleiben selbständig nutzbar. Zugleich werden ungeeignete Eingriffe in diese Bausteine vermieden. Die genannten Grundsätze machen dafür aber i.a. den Einsatz von speziellen Kopplungsmoduln notwendig. Diese werden in Assembler geschrieben und sorgen für einen reibungslosen Informationsaustausch zwischen nichtkompatiblen Programmbausteinen: Bestimmte Moduln werden nicht direkt aufgerufen, sondern über den Umweg eines 'Adapters'. Damit wird auch ein Zusammenwirken von Bausteinen möglich, die in verschiedenen höheren Programmiersprachen realisiert sind.

Programmiersprache

Die Hauptbausteine von SYSDFV sind in FORTRAN IV geschrieben. Darüber hinaus werden für die Durchführung von Elementaroperationen eine ganze Reihe von Assembler-Moduln eingesetzt.

Verwendete Lösungsbausteine

Zur Lösung parabolischer Differentialgleichungen stehen verschiedene Differenzenverfahren (sowohl explizite als auch implizite) zur Verfügung. Die bisher vorhandenen Lösungsbausteine sind speziell auf das Steuerprogramm zugeschnitten. Künftig sollen Verfahren hinzu kommen, die auf der Semidiskretisierung beruhen. Dabei wird auf vorhandene Bausteine zur Lösung von gewöhnlichen Differentialgleichungssystemen zurückgegriffen werden.

Zur Lösung von Zwei-Punkt-Randwertproblemen wird auf vorhandene Standardsoftware auf diesem Gebiet zugegriffen.

SYSDFV gewährleistet einen reibungslosen Übergang vom Anfangs-Randwertproblem (parabolische Differentialgleichung) zum Randwertproblem und umgekehrt.

Literaturhinweis

/1/ Böhme,P.: Mathematische Untersuchungen zum dynamischen Verhalten eines chemischen Rohrreaktors , THLM 1982

Verfasser:
Dipl.-Math. Peter Böhme
TH 'Carl Schorlemmer' Leuna-Merseburg
Organisations- und Rechenzentrum
4200 Merseburg, Geusaer Str.

Fenske, Bernd und Meißler, Hans-Günter

"Ein Konzept zum Einlesen eines Stromes von Lochbanddateien"

## 1. Einleitung

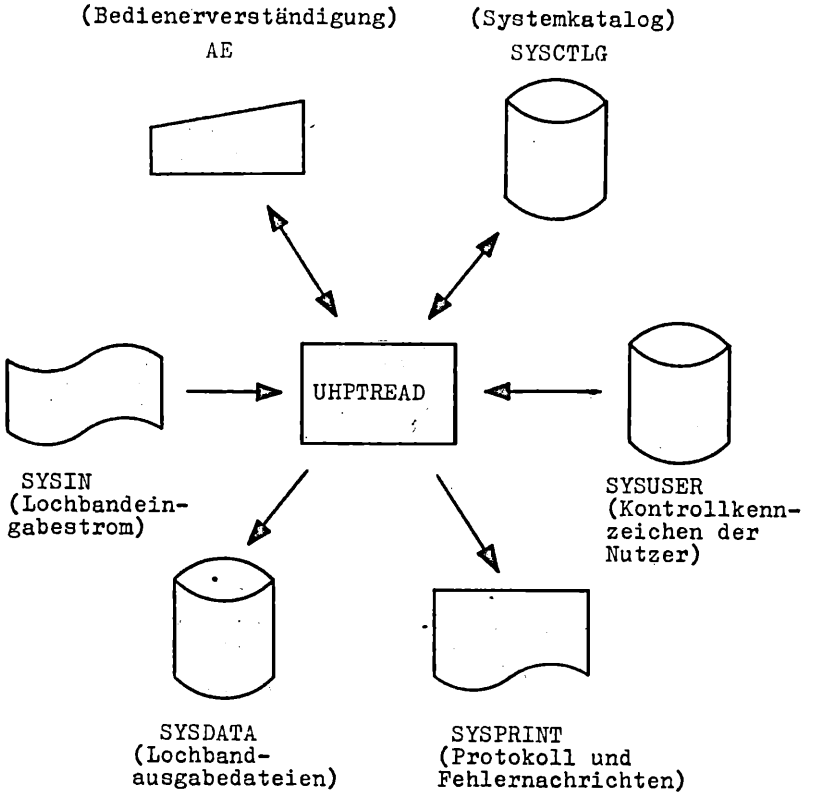
Auch an modernen ESER-Rechenanlagen hält der Bedarf zur Eingabe von LB-Dateien unvermindert an. Gründe dafür liegen einerseits in der nur zögernden Ablösung des Lochstreifens durch alternative Datenträger, wie z. B. Magnetkassetten, bzw. durch eine Online-Datenerfassung über Terminals und andererseits in dem Fortbestehen der traditionellen Datenerfassung auf der Basis von Lochstreifengeräten. Demgegenüber steht die Tatsache, daß das Einlesen von LB-Dateien durch das Betriebssystem OS/ES nur sehr mangelhaft unterstützt wird. Neben einem hohen Bedienaufwand, der in sich bereits viele Fehlermöglichkeiten birgt, wird die LB-Eingabe vor allem durch die Unsicherheit der Lesegeräte und durch einen unzureichenden Service des Betriebssystems in Fehlersituationen (Prüfbitfehler, Gerätefehler) behindert. Im folgenden wird das Konzept des Programmes UHPTREAD vorgestellt, das am ORZ der MLU Halle entwickelt wurde, um diesen Problemen wirksam zu begegnen.

## 2. Funktionsprinzip von UHPTREAD

Als Grundprinzip für die Arbeit von UHPTREAD wird eine Trennung der Dateieingabe von der eigentlichen Dateiverarbeitung angestrebt. Alle LB-Dateien, die in einem bestimmten Zeitraum zur Verarbeitung anfallen, werden mit einem Aufruf von UHPTREAD nacheinander eingelesen und für die spätere Verarbeitung auf DA-Datenträgern geeignet zwischengespeichert. UHPTREAD empfängt also eingabeseitig einen Strom von LB-Dateien. Das Programm wird nicht vom Nutzer selbst aufgerufen, sondern als Systemprogramm bei Bedarf vom Bediener gestartet. Erst nachdem der Einlesevorgang beendet ist, werden die Nutzerjobs vom Bediener freigegeben,

welche dann auf die zwischengespeicherten LB-Dateien zugreifen. Abbildung 1 zeigt die Dateien bzw. externen Geräte, mit denen UHPTREAD arbeitet.

Abb. 1




## 3. Erzeugung der LB-Ausgabedateien

Für jede eingelesene LB-Datei im Eingabestrom wird von UHPTREAD eine sequentielle LB-Ausgabedatei erzeugt. Diese wird vom Programm selbst im VTOC eines geeignet gewählten DA-Datenträgers mittels SVC 32 (ALLOCATE) eingerichtet und katalogisiert. Als Vorbild diente hier die Arbeitsweise des Systemeingabeprogramms

(vgl. DD-Anweisung IEFDATA beim Systemeingabeprogramm). Die LB-Ausgabedateien erhalten einen charakteristischen mehrstufigen Namen, anhand dessen sie vom Nutzerjob zur Verarbeitung wieder aufgefunden werden können. Dieser Dateiname besteht aus drei Teilen

DSN = SYSP.username.tapename

Der erste Teil enthält ein festes Kennzeichen (SYSP). Durch den zweiten Teil wird der jeweilige Nutzer identifiziert, der diese Datei verarbeiten will. Mit Hilfe einer Datei SYSUSER (vgl. Abb. 1) ist es möglich, über eine EXIT-Routine diese Nutzerkennzeichen zu überprüfen und dadurch abzusichern, daß sich unterschiedliche Nutzer nicht gegenseitig stören können. Es ist z. B. sinnvoll, an dieser Stelle den Namen des Jobs angeben zu lassen, der die LB-Ausgabedatei verarbeiten soll. Der dritte Teil kann vom Nutzer frei gewählt werden und dient ihm zur Identifikation seiner jeweiligen Lochbanddatei. Die LB-Ausgabedateien haben ungeachtet ihres Namens grundsätzlich temporären Charakter. Das heißt, sie werden im allgemeinen nur bis zum Lauf des verarbeitenden Jobs aufgehoben. Wird beim Einrichten bereits eine gleichnamige Datei bzw. eine Katalogeintragung vorgefunden, werden diese als verfallen gewertet und gelöscht.

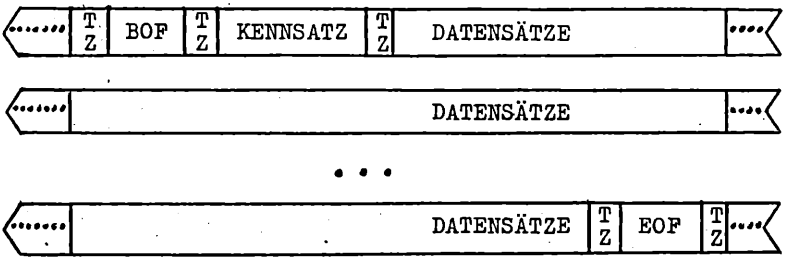
## 4. Struktur der LB-Eingabedateien

Das Einlesen des Eingabestromes erfolgt durch UHPTREAD unabhängig von der Struktur der einzelnen Dateien mit einer festen, großen Satzlänge. Die Aufspaltung in einzelne Sätze und die Konvertierung in den DKOI-Code erfolgt erst im Hauptspeicher durch das Programm selbst. Dadurch ist die Möglichkeit einer gewissen Bearbeitung der Sätze gegeben. So kann z. B. das Satzformat entsprechend wahlfreien Parametern der LB-Ausgabedatei geändert werden.
Die Struktur der LB-Eingabedateien ist so vorgeschrieben, daß eine eindeutige Unterscheidung zu vorangehenden und nachfolgenden Daten im Eingabestrom besteht. Zu diesem Zweck müssen die Dateien durch bestimmte, vom Lochbandcode abhängige Beginn- und Endemarkierungen, begrenzt sein. Alle Identifikationsanga-

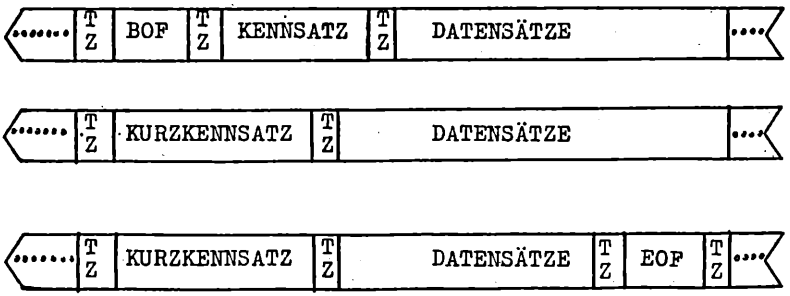
ben und Parameter zur Beschreibung der Eingabedatei, zur Satzbearbeitung und für die Ausgabedatei sind in einem speziellen Kennsatz, der unmittelbar nach der Beginnmarkierung folgt, anzugeben.

Abbildung 2 zeigt den Aufbau einer LB-Eingabedatei.

A) Lochbanddatei ohne Reihenfolgeprüfung der Lochstreifen:




B) Lochbanddatei mit Reihenfolgeprüfung der Lochstreifen:




TZ - Trennzeichen, vom LB-Code abhängig

## 5. Dienstleistungen des Programmes

Fólgende Dienstleistungen werden von UHPTREAD standardmäßig bzw. über Parameteranforderung im Kennsatz realisiert:

- Katalogisieren der Ausgabedatei

- Prüfung auf Zulässigkeit zur Bearbeitung anhand der Nutzerkennzeichen und einer Nutzerdatei
- Satzformat F und U eingabeseitig
- Automatische Satzendeerkennung (EORC)
- Wahlweise Übersetzung in den DKOI-Code (es werden alle LB-Codes des OS/ES unterstützt)
- Unterschiedliche LB-Codes für Kennsatz und für Datensätze möglich
- Ersetzen von fehlerhaften Zeichen (ungültig oder paritätsfalsch) durch ein Ungültigkeitszeichen
- Umwandeln des Satzformates und der Satzlänge ausgabeseitig (zu lange Sätze werden abgeschnitten, zu kurze Sätze durch ein Füllzeichen aufgefüllt)
- Ausblenden von Transportlochungen und Irrungszeichen
- Ausblenden von Sätzen, die ein Satzirrungszeichen enthalten
- Wahlweise Reihenfolgeprüfung (incl. Identitätsprüfung) der einzelnen Lochstreifen einer LB-Datei
- Wahlweise Umwandlung von Kleinbuchstaben in Großbuchstaben bei der Übersetzung in den DKOI-Code
- Wahlweise Verarbeitung von Tabulatoren (max. bis Satzposition 99)
- Wahlweise Bereitstellung eines Fehlerbeschreibungswortes in jedem Satz (verlängert, verkürzt, ungültige Zeichen)
- Standardmäßige Bereitstellung von Informationen über das Einlesen der Datei im Benutzervorsatz UHL1 der Ausgabedatei

Die Angaben im Kennsatz werden in Form von Stellungsparametern (Identifikationsangaben) und Kennwortparametern (wahlfreie Funktionen) gemacht. Die Abbildung 3 zeigt den Aufbau des Kennsatzes.

Abb. 3 Kennsatz einer LB-Datei

| | | |
|---|---|---|
| {,c} | Kennsatzcode | |
| {,username} | Nutzerkennzeichen | 1) |
| {,tapename} | Lochbandname | |
| [,number] | Lochstreifennummer | |
| [,INPUT={F/U}] | Satzformat-Eingabe | |
| [,CODE={A/B/C/F/I/K/L/M/N/P/Q/T/U}] | Datencode | 2) |
| [,EORC=hh] | Satzendekennzeichen | |
| [,BUFL=nnnnn] | Satzlänge-Eingabe | |
| [,DELETEC=hh] | Irrungszeichen | |
| [,DELETER=hh] | Satzirrungszeichen | |
| [,WRONGC=hh] | Ungültigkeitszeichen | |
| [,FILLC=hh] | Füllzeichen | 3) |
| [,CHANGE={YES/NO}] | Umwdlg. in Großbst. | |
| [,TABC=hh] | Tabulatorzeichen | |
| [,TABSET=n1n2...nk] | Tabulatorpositionen | |
| [,RECFM={U/F/FB/FBS/V/VB/VS/VBS}] | Satzformat- Ausgabe | |
| [,LRECL=nnnnn] | Satzlänge - Ausgabe | 4) |
| [,BLKSIZE=nnnnn] | Blocklänge - Ausgabe | |
| [,ERRORDW={YES/NO}] | Fehlerbeschreibg.Wort | |

1) - Identifikationsangaben
2) - Beschreibung der Eingabedatei
3) - Beschreibung der Satzaufbereitung
4) - Beschreibung der Ausgabedatei

## 6. Hilfsprogramme

Für die praktische Handhabung von UHPTREAD im Rechenbetrieb existieren folgende Hilfsprogramme:

SHSYSP - Löschen aller verfallenen LB-Ausgabedateien auf allen aufliegenden DA-Datenträgern

SFLBTEST - Test des Lochbandlesers auf Funktionstüchtigkeit
SIPRINT - Erzeugen einer druckaufbereiteten Tabelle eines Lochbandcodes aus einer vorliegenden maschineninternen Übersetzungstabelle
UHL1READ - UP. zum Einlesen des von UHPTREAD erzeugten UHL1 für ASSEMBLER- und FORTRAN-HP.
UHL1XPL - analog für PL1-HP. (normaler und optimierender Compiler)

## 7. Praktische Erfahrungen bei der Anwendung von UHPTREAD

UHPTREAD wird seit ca. 1 1/2 Jahren am ORZ der MLU mit Erfolg angewendet. Die damit verbundene Technologie der Lochbandeingabe wurde für alle Nutzer zur Pflicht gemacht. Auch große Projekte, die auf Lochstreifenbasis arbeiten (LIS) sind auf diese Technologie übergegangen. Das Erlernen der Steuersprache von UHPTREAD erfordert zwar seitens der Nutzer einen geringen Aufwand, der aber durch die damit verbundenen Dienstleistungen belohnt wird. Wesentlich wurde die Arbeit der Bediener beim Einlesen von LB-Dateien erleichtert.
Der Funktionsumfang des Programms wird unterschiedlich genutzt. Häufig gebraucht werden die Umwandlung des Satzformates (geblockte Ausgabe), die Anwendung von Irrungs- und Satzirrungszeichen, das Umwandeln von Klein- in Großbuchstaben und das Einlesen (und Übersetzen) von fehlerbehafteten Lochbändern mit Ersetzen der ungültigen Zeichen durch das Ungültigkeitszeichen. Sehr selten genutzt werden dagegen die Möglichkeiten der Bereitstellung von Fehlerinformationen (UHL1, ERRORDW), die Verarbeitung von Tabulatoren und die Reihenfolgeprüfung der einzelnen Lochstreifen einer LB-Datei.
Aus dem praktischen Betrieb haben sich ferner folgende Wünsche für die Erweiterung des Funktionsumfangs von UHPTREAD ergeben.

- Angabe einer beliebigen Menge von Irrungszeichen, die beim Einlesen der Daten auszublenden sind, statt wie bisher nur eines Irrungszeichens.
- Verarbeitung eines Kontrollstreifens zur Prüfung der

Funktionssicherheit des LB-Lesers (Integration der Funktion des Hilfsprogramms SFLBTEST in UHPTREAD selbst).

- Vermittlung des Ausschnittes der SYSPRINT-Datei, der eine bestimmte LB-Datei betrifft, an den verarbeitenden Job. Zur Zeit ist es nur möglich, das Protokoll von UHPTREAD bei der Rückgabe der Jobs vom Nutzer einsehen zu lassen.
- Bequemes Einbringen von neuen (privaten) LB-Codes in die Verarbeitungsmoduln von UHPTREAD. Zur Zeit sind nur die im OS/ES standardisierten LB-Codes vorhanden.

An der Realisierung dieser Forderungen wird gegenwärtig gearbeitet.

Verfasser: Dipl.-Math. B. Fenske, Dipl.-Math. H.-G. Meißler

Organisations- und Rechenzentrum der
Martin-Luther-Universität Halle-Wittenberg
4020 Halle/S., Weinbergweg 17

Hagen, Manfred
Klaembt, Egbert
Schirmbacher, Peter

# Methodische Auswahl eines Datenauswertungssystems für den Einsatz in einer Informationszentrale

## 1. Einleitung

Der Anteil der nichtschematischen Aufgaben innerhalb der Datenverarbeitungsaufgaben auf dem Gebiet der Leitungsunterstützung nimmt ständig zu. Nichtschematische Aufgaben sind nicht langfristig vorhersehbare, relativ einmalige und kurzfristig zu lösende Datenverarbeitungsaufgaben. Sie lassen sich durch die herkömmliche Einzelprogrammierung nicht sinnvoll lösen. Einen gangbaren Weg sehen wir in der Arbeit einer Informationszentrale, wobei die Nutzung von Datenauswertungs- und -verwaltungssystemen (DAVS) aus unserer Sicht eine Alternative zu Datenbanksystemen sein kann.
Unter einem DAVS verstehen wir eine einheitliche Datenbasis (ein Dateiensystem), für das vorgefertigte Programme und organisatorische Regelungen zu ihrer Verwaltung existieren und ein Datenauswertungssystem (DAS) zu ihrer Auswertung. Ein DAS ist ein System von Softwareprodukten mit möglichst hohem Automatisierungsgrad, das zur Lösung nichtschematischer Aufgaben durch Auswertung einer einheitlichen Datenbasis eingesetzt wird. Das Spektrum der Softwareprodukte, die für bzw. als DAS im Sinne dieser allgemein gehaltenen Definition genutzt werden können, ist recht umfangreich. Dazu zählen u.E. Generatoren, Universalprogramme, Makrosysteme und Abfragesprachen. In der augenblicklichen Situation stehen nachnutzbar vor allem Generatoren und interpretierende Systeme zur Verfügung, wobei deren Anzahl ständig zunimmt.

Die Lösung nichtschematischer Aufgaben ohne Benutzung von DBS wird eine wichtige Rolle spielen, zumindest für die nächsten Jahre auf Grund der gegenwärtigen Datenbankproblematik, wie Prof. Schoppan auf dem 10. Kolloquium über Rechentechnik und Da-

tenverarbeitung in Magdeburg sagte, und, wie wir glauben, auch darüber hinaus in Abhängigkeit von den konkreten Einsatzbedingungen. Demzufolge gewinnen die Probleme des Einsatzes, der Bewertung und der Auswahl von DAS an Bedeutung. Für jeden Nutzer, der seine nichtschematischen Aufgaben mit DAS lösen will, steht die Frage, welches konkrete DAS für seinen konkreten Anwendungsfall das geeignetste ist. Es handelt sich hierbei um Fragen der Auswahl bzw. Selektion von Software. Damit sind wir bei einem allgemeinen Problem. Natürlich wurde Software seit langem in der einen oder anderen Weise ausgewählt, nur meist nicht sehr methodisch. Die Entwicklung in der EDV geht aber zur verstärkten Nachnutzung von Software bzw. zur immer umfangreicheren Produktion von extra zur Nachnutzung hergestellten Software, damit auch zu umfangreicherer Softwareselektion. Wie auf anderen Gebieten der EDV (z.B. Programmierung, Entwurf) wurden hierzu in der Literatur der letzten Jahre eine Reihe von teilweise recht umfangreichen Methoden vorgestellt. Uns geht es darum, die allgemeinen Methoden auf die Auswahl von DAS anzuwenden und eine auf diesen Fall bezogene praktische und relativ einfache Variante anzugeben. Dafür sind einige Kompromisse notwendig, auf die wir an gegebenerStelle hinweisen.

## 2. Softwareselektion

Zur Softwareselektion gehören eigentlich zwei Entscheidungen. Zum einen, ob Nachnutzung oder Eigenerstellung erfolgen soll, und zum anderen, wenn Nachnutzung die Auswahl aus mehreren Produkten. Wir beschränken uns im weiteren auf die Entscheidung für Nachnutzung. Das scheint gerechtfertigt, da

1. die Nachnutzung die vorzuziehende Lösung ist,
2. für unsere Auswahl genügend nachnutzbare Produkte zur Verfügung stehen,
3. die wenigsten Einrichtungen zur Eigenerstellung die Möglichkeiten besitzen.

Softwareselektion ist ein subjektiver Bewertungsprozeß, zu dessen Durchführung geeignete Methoden herangezogen werden. Wesentlich ist, daß Softwarebewertung immer von den konkreten Anforderungen des Anwenders abhängt, d.h. es geht nicht um das beste

Produkt schlechthin, sondern um das für den Anwendungsfall am besten geeignete. Die Softwareselektion umfaßt nach Frank /2/ folgende organisatorische Phasen:

1. Bestimmung des Anforderungsprofils
2. Sammlung von Produktinformationen und Feststellung des Leistungsprofils
3. Produktvergleich und Auswahlentscheidung (Softwareselektion i.e.S.)
4. Vertragsverhandlung, Implementierung, Effizienzprüfung

Dabei sind diese Phasen nicht isoliert voneinander, vielmehr überschneiden sie sich und laufen auch teilweise iterativ ab. Wir können hier nur schwerpunktmäßig und kurz auf die einzelnen Phasen eingehen.

## 2.1. Bestimmung des Anforderungsprofils

Das Anforderungsprofil ist die Grundlage für die Auswahlentscheidung (sie ist übrigens auch Grundlage für die Eigenerstellung von Software), denn Versäumnisse, Fehler und Ungenauigkeiten führen nahezu zwangsläufig zu nachteiligen Folgen beim Einsatz der Software. In unserem Fall muß die Bestimmung des Anforderungsprofils letztlich zu einem konkreten auf die DAS-Auswahlentscheidung zugeschnittenen Kriterienkatalog führen. Es ist jedoch in dieser Phase nicht notwendig, den vollen Detaillisierungsgrad zu erreichen, denn die folgenden Phasen sollen in die Konkretisierung mit einfließen. Zum Inhalt des Anforderungsprofils für ein DAS gehören u.E. folgende Hauptpunkte:

1. die erforderlichen Leistungen des DAS
2. die Erfordernisse der vorhandenen Hard-/Software
3. Handhabung und Kosten

Sowohl unbedingt notwendig als auch wünschenswerte Eigenschaften sollten als solche gekennzeichnet werden. Vor der Angabe eines möglichen Kriterienkatalogs gehen wir noch auf die nächsten Phasen ein.

## 2.2. Sammlung von Produktinformationen und Feststellung des Leistungsprofils

Die Sammlung von Produktinformationen erfolgt günstig nicht erst nach der Bestimmung des Anforderungsprofils, sondern möglichst kontinuierlich. Es zeigt sich nämlich, daß das gar nicht so einfach ist, denn ein Katalog mit detaillierten Angaben über angebotene Softwareprodukte existiert nicht, wäre aber wünschenswert. Man ist also angewiesen auf Informationen, die in Fachzeitschriften, auf Kongressen, in Kundeninformationen, über die Projekt- und Programmzentrale, über das ZIID oder mündlich zu erhalten sind. Sobald mögliche Produkte und Hersteller bekannt sind, kommt man nahtlos zur Bestimmung des Leistungsprofils. Der Interessent kann sich dabei auf Informationen des Herstellers, Erfahrungen bisheriger Nutzer und eventuell auf Probeinstallationen stützen. Es scheint aber auf jeden Fall geraten, sowohl die vom Hersteller als auch die von bisherigen Nutzern erhaltenen Informationen kritisch zu betrachten.
Nach dem oben schon erwähnten Frank sollten vom Hersteller zumindest folgende Angaben vorliegen:

- die Dokumentationsunterlagen
- Software/Hardwareerfordernisse (auch für Installation)
- verwendete Methoden
- Datum der Erstanwendung
- bisherige Anwender (Anzahl)
- Erfahrungen beim Einsatz
- Möglichkeiten der Einführungsunterstützung bzw. Schulung
- Zusatzleistungen, geplante Ausbaustufen
- Kosten

Empfehlenswert ist sicher eine Probeinstallation, aber sie wird auf Grund des Aufwandes wohl nur selten möglich sein. Das Leistungsprofil ergibt sich aus allen verfügbaren Informationen. Als Beispiel möge die Kurzfassung des Leistungsprofils von ATLAS dienen.

Leistungsprofil ATLAS (I) (Kurzfassung)

Hersteller: LfA Berlin (A. Gustavs)
Kurzcharakteristik: Programmgenerator, compilierend, aus Fragenkatalog und Nutzerexits generiertes PLA-Programm
Anzahl der Installationen: ungefähr 50
BS, Konfiguration: OS/ES, 1 WPS, 1 LKL, PD
Schulungsmöglichkeit: bei WBA DVZ (1 Woche)
Weiterentwicklung: ATLAS II (existiert) vorgesehen: interaktive Erstellung, DB-Anbindung
Leistungen: 9 sequentielle Eingabedateien
3 Ausgabedateien, 3 Listen

Aus dem Vergleich von Anforderungsprofil mit den Leistungsprofilen erkennt man, welche Softwareprodukte für die Auswahl in Frage kommen. Das Vorhandensein geeigneter Software setzen wir voraus, ansonsten bleibt nur Eigenerstellung oder Änderung der Anforderungen. Eigentlich ist an dieser Stelle zu klären, ob Eigenerstellung oder Nachnutzung sinnvoll ist, dieses soll aber, wie vorhin schon ausgeführt, zugunsten der Nachnutzung hier nicht diskutiert werden. Damit kommen wir zum Kernstück der Softwareselektion.

## 2.3. Auswahlentscheidung

Gesucht ist in unserem Fall im DAS Pi aus der Menge der zulässigen DAS $P_1,\ldots,P_n$ . Wir orientieren dabei auf ein einfach handbares Matrixverfahren aus der Gruppe der mehrdimensionalen Auswahlverfahren. Allgemein formuliert ordnen die Matrixverfahren, die auf einem Kriterienkatalog basieren, jedem Softwareprodukt eine Zahl zu. Als bestgeeignetes Produkt wird das ausgewählt, das die größte Zahl erhält. Diese Verfahren erfordern natürlich, daß jedem Produkt bzgl. jedes Kriteriums eine Zahl zugeordnet wird. Damit dies sinnvoll erfolgen kann, muß für die möglichen Merkmale der Kriterien eine Vergleichsrelation vorliegen. Das bedeutet, diese Verfahren setzen die Existenz von Verhältnisskalen (zumindest Intervallskalen) zur Bewertung der

Softwareprodukte bzgl. der Kriterien voraus. Nun ist diese Voraussetzung alles andere als selbstverständlich. Kriterienkataloge sind in der EDV schon oft in Erscheinung getreten, vor allem bei der Bewertung von Datenbanksystemen (z.B. von Schubert/ Belke TU Dresden /1/). Dabei wird zur Bewertung der Merkmale innerhalb der Kriterien vielfach von Äquivalenzrelation bzw. Halbordnungsrelation ausgegangen, was dann bei der Auswahlentscheidung zu Rangordnungsverfahren führen kann. Trotz unseres einfacheren Auswahlproblems spielen solche Kriterien ebenfalls eine Rolle. Wir können diese natürlich nicht einfach weglassen. Wir glauben aus praktischen Erwägungen heraus jedoch, daß es hier möglich ist, künstlich Verhältnisskalen einzuführen und mit kleinen Ergänzungen zum Matrixverfahren doch noch sinnvolle Ergebnisse zu erhalten.
Kriterien, die nicht skalierbar sind, können nicht in den Katalog aufgenommen werden.
Die Einführung von künstlichen Verhältnisskalen bedeutet z.B. für Ja-Nein-Kriterien, daß Nein Ø und Ja der höchste Wert zugeordnet wird.
Allgemein sollen die Kriterien weitestgehend unabhängig und objektiv meßbar sein und der Kriterienkatalog alle relevanten Dimensionen umfassen, aber nicht zu umfangreich sein.

Unser Vorschlag für einen Kriterienkatalog hat folgendes Aussehen:

Kriterienkatalog (DAS)

| Kriteriengruppen | Kriterien |
|---|---|
| K1. Systemtechn. Teil | K1.1. HS-Bedarf<br>K1.2. BS<br>K1.3. Laufzeitverhalten<br>K1.4. Peripherie<br>K1.5. DAS-Bibliotheken |
| K2. Handhabung | K2.1. Erlernbarkeit<br>K2.2. Dokumentation<br>K2.3. Text<br>K2.4. Automatisierungsgrad<br>K2.5. organisatorische Flexibilität |
| K3. Hersteller | K3.1. Herstellerqualifikation<br>K3.2. Anzahl der Installationen<br>K3.3. Vertragsform |
| K4. Systemkonzept | K4.1. DAS-Konzept<br>K4.2. Sprachkonzept<br>K4.3. Qualität |

| | |
|---|---|
| K5. Kosten und Zusatzleistungen | K5.1. Kosten<br>K5.2. Wartung<br>K5.3. Einführungsunterstützung/Schulung<br>K5.4. Weiterentwicklung |
| K6. Leistungen | K6.1. Eingabedateien<br>K6.2. Anzahl der Eingabedateien<br>K6.3. Verknüpfungskomfort<br>K6.4. Dateibeschreibung<br>K6.5. Ausgabedateien<br>K6.6. Arithmetische Verknüpfungen<br>K6.7. Gruppenbehandlung<br>K6.8. Sortierung<br>K6.9. Selektion<br>K6.10. Schlüsselverarbeitung<br>K6.11. Druckaufbereitung<br>K6.12. Verarbeitungsarten |

Die Kriterien sind zu Kriteriengruppen zusammengefaßt. Die einzelnen Kriterien werden innerhalb der Kriteriengruppen und die Kriteriengruppen zueinander gewichtet. Das ergibt sich daraus, daß man vollständige Unabhängigkeit der Kriterien nicht erreichen kann und die Kriteriengruppen bzw. die Kriterien innerhalb der Gruppen unterschiedlich für die Auswahl ins Gewicht fallen. Welche Kriterien in den Kriterienkatalog aufgenommen und wie die Gewichte verteilt werden, ist abhängig von den konkreten Anforderungen des Nutzers. Kriterien, die sich auf unverzichtbare Eigenschaften eines DAS aus Nutzersicht beziehen, werden extra gekennzeichnet, sie spielen später noch eine wichtige Rolle. Es ist sicher sinnvoll für die Bewertung bzgl. eines Einzelkriteriums, vorher schon Überlegungen, bzw. wie welche Merkmale bewertet werden sollen, zu fixieren (siehe Beispiel 1).

| Beispiel 1: | Kriterien | Einzelüberlegungen |
|---|---|---|
| | K6.1. Eingabedateien | Satzformate<br>Datenformate<br>Einschränkungen |

Falls eine Probeinstallation möglich war, ist die Bewertung wichtiger Kriterien (z.B. Laufzeit, HS-Bedarf) auch als Ergebnis von eindimensionalen Verfahren (z.B. Benchmarktest) zu erhalten. Dieser Prozeß der Auswahl und Gewichtung der Kriterien sowie die Bewertung der Produkte bzgl. der Einzelkriterien ist der die Auswahl entscheidende Prozeß, deshalb sollte er äußerst sorgsam und sehr kritisch durchgeführt werden.

Als Zwischenergebnis erhalten wir eine Matrix, in der jede Zeile ein Kriterium, jede Spalte ein DAS und jedes Matrixelement die Bewertung eines DAS bzgl. eines Kriteriums darstellt.
Im einfachen Beispiel 2 ist eine solche Matrix angegeben und wie das von uns gewählte Verfahren arbeitet.

Beispiel 2: Gewichtsstufenmethode

| Gestufte Kriterien | Gewichtung | | Noten (0.0–10.0) | | | Bewertungszahl | | |
|---|---|---|---|---|---|---|---|---|
| | | | Produkte | | | Produkte | | |
| | Stufe 1 | Stufe 2 | A | B | C | A | B | C |
| K1.1. | 10 | | 3 | 8 | 10 | 30 | | 100 |
| K1.2. | 30 | | 5 | 0 | 6 | 150 | | 180 |
| K1.3. | 60 | | 7 | 2 | 2 | 420 | | 120 |
| | | | | | | 600 | | 400 |
| K1 | 100 | 0.7 | | | | 420 | | 280 |
| K2.1. | 20 | | 5 | 8 | 6 | 100 | | 120 |
| K2.2. | 20 | | 7 | 9 | 5 | 140 | | 100 |
| K2.3. | 10 | | 6 | 10 | 3 | 60 | | 30 |
| K2.4. | 50 | | 6 | 10 | 9 | 300 | | 450 |
| | | | | | | 600 | | 700 |
| K2 | | 0.3 | | | | 180 | | 210 |
| Gesamtbewertung | | 1 | | | | 600 | | 490 |

Zuerst werden die besonders gekennzeichneten Kriterien, die unverzichtbare Eigenschaften betreffen, behandelt. Hat ein DAS eine Ø erhalten, fällt es aus dem weiteren Auswahlprozeß heraus. Deshalb werden sie auch als KO-Kriterien bezeichnet. Das benutzte Verfahren ist die Gewichtsstufenmethode (zweifache Gewichtung). Bei vorliegender Bewertungsmatrix ist nur noch eine einfache Rechnung notwendig. Das DAS mit der höchsten Punktzahl ist das ausgewählte (im Beispiel 2 das Produkt A).
Dem schließen sich noch einige Arbeiten, so bei knappen Ergebnissen eine Verfeinerung, grafische Darstellungen (z.B. Spinne) und eine Effizienzprüfung an, auf die wir hier nicht weiter eingehen wollen, die aber, wie auch die Vertragsverhandlung, das Auswahlergebnis noch beeinflussen können.

## 3. Schlußbemerkung

Wir sind uns bewußt, daß das von uns hier angegebene Matrixverfahren durchaus problematisch ist, vor allem hinsichtlich der Notwendigkeit von Verhältnisskalen. Unser Ziel bestand zum einen darin, auf Möglichkeit und zunehmende Notwendigkeit von methodischer Softwareauswahl gerade auf diesem von uns für wichtig gehaltenen Gebiet hinzuweisen, und zum anderen darin, einen praktischen Vorschlag für eine relativ einfach handhabbare Selektion von DAS zu unterbreiten. Wir denken, daß eine gut gelungene Softwareselektion dem Nutzer einige Probleme ersparen kann. Wir glauben auch, daß eine breite Anwendung dieser Methoden zu Verbesserungen auf der Herstellerseite führen kann, in der besseren Bereitstellung von für die Nutzer wichtigen Produktinformationen und letztlich in der Qualität der Softwareprodukte.

## Literaturhinweise


/1/ Belke, Wolfgang "Bewertung von DBBS durch Klassifizierung" in RT/DV, Heft 4, 1982

/2/ Frank, Joachim "Standard-Software", Köln 1977


## Verfasser


Dipl.-Math. Manfred Hagen
Dipl.-Ing. Egbert Klaembt
Dipl.-Wirt. Peter Schirmbacher
ORZ der Humboldt-Universität zu Berlin
1086 Berlin
PSF 1297

Wittig, Wolfram

## Elemente eines Planungsdialoges im Fernmeldewesen

### 1. Technologische und ökonomische Bedingungen des Reproduktionsprozesses im Fernmeldewesen als Basis einer Modellkonzeption zur Vorhersage des Bedarfs an Ausrüstungen und Material

#### 1.1. Bedingungen des technologischen Prozesses im Fernmeldewesen

Eine Erhöhung des "Produktionsausstoßes" im Fernmeldewesen, der örtlich veränderten Nachricht, zur umfassenden Befriedigung der Nachrichtenverkehrsbedürfnisse der sozialistischen Gesellschaft ist nur möglich, wenn ein reibungsloses und proportionales Zusammenspiel der wesentlichen technisch-technologischen Komponenten zur Realisierung des Fernmeldeverkehrs gesichert ist. Bezogen auf den bereitzustellenden Ausrüstungs- und Materialanteil bedeutet dies, die Sicherung einer proportionalen technologisch wohl begründeten Vorhersage des Bedarfs an Endeinrichtungen des Fernsprech-, Fernschreibverkehrs und der Datenübertragung, der Vermittlungstechnik, der Übertragungstechnik und des Bedarfs an Kabeln und Leitungen für den Ausbau und die Instandhaltung der Netze. Weitere zu berücksichtigende Merkmale sind die Struktur des Netzes mit den Komponenten Fernsprech-, Fernschreib-, Datenübertragungs-, Rundfunknetz und die existierenden Netzebenen insbesondere die Unterteilung in Ortsnetz- und Weitnetzebene.

Als Materialverbraucher tritt für den Ausbau des Fernmeldenetzes vorrangig die Produktionskapazität des Sprechstellenbaus und des Fernmeldebaus und für die Instandhaltung die des Technischen Dienstes in Erscheinung.

#### 1.2. Eine ökonomische Kausalkette für die Planung des Reproduktionsprozesses im Fernmeldewesen

Jeder Entwurf eines Modells zur Planung des Reproduktionsprozesses im Fernmeldewesen wird bestimmt von den Planungszielen, die mit dem Modell unterstützt werden sollen und von der grundsätzlichen "Modellphilosophie", die der Autor zu Grunde legt. Ein Prognose- und Planungsmodell für den Bedarf an Ausrüstungen und Material im Fernmeldewesen muß insbesondere für mittel- und lang-

fristige Zeiträume, auf Grund der beschränkten apriori Kenntnisse über z.B. im Zeitraum von 5 Jahren zu realisierende Maßnahmen und Vorhaben und ihrer Konsequenzen für den Ausrüstungs- und Materialbedarf, aber auch auf Grund der real existierenden komplizierten Verflechtungen zwischen den Vorhaben verschiedener technologischer Bereiche, den Versuch unternehmen, objektiv vorhandene ökonomische Kausalstrukturen zu implementieren.

Damit kommt man zwangsläufig zu einem komplexen Gesamtmodell für den Reproduktionsprozeß, das es mehr oder weniger grob gestattet, ausgehend von den Zielformulierungen der Nachrichtenverkehrsbedürfnisse der Gesellschaft die Konsequenzen für die Grundfondsreproduktion, die Leistungsentwicklung im Fernmeldebau, Technischen Dienst und Sprechstellenbau sowie letztlich für den Bedarf an Ausrüstungen und Material sichtbar zu machen. Die daraus resultierende und in Abbildung 1 dargestellte Grundstruktur für den Modellansatz beschreibt den Planungsverlauf.

$$K_t \text{ ----- } GF_{t-r} \text{ ----- } L_{t-r-s} \text{ ----- } M_{t-r-s}$$

$K_t$ - Zuwachs an technologischen Kapazitäten des Fernmeldewesens (Zuwachs an Hauptanschlüssen, Zuwachs an Anschlußmöglichkeiten in Vermittlungsstellen etc.) im Jahre t

$GF_{t-r}$ - Notwendiger Zuwachs an technischen Grundfonds des Fernmeldewesens im Jahre t-r

$L_{t-r-s}$ - Notwendige Produktionskapazitäten im Jahre t-r-s

$M_{t-r-s}$ - Relevanter Bedarf an Ausrüstungen und Material im Jahre t-r-s

Abb. 1

## 2. Datenfonds, Methodenfonds und Technologie des Analyse- und Planungsinstrumentariums für das Fernmeldewesen

### 2.1. Datenfonds

Die Möglichkeiten von Verfahren zur Unterstützung des Analyse- und Planungsprozesses werden von Umfang und Konsistenz der zur Verfügung stehenden Datenfonds, von der Algorithmenbasis, die zur Verarbeitung dieser Datenfonds verfügbar ist und von der technologischen Gestaltung des variablen Zugriffs zu Daten- und Methodenfonds bestimmt.

Die Datenfonds des hier diskutierten Verfahrens bestehen aus Quartalsdaten für 4265 Kennziffern des Reproduktionsprozesses Fernmeldewesen für die Bereiche Kapazitätsentwicklung, Grundfondsentwicklung, Entwicklung produktiver Leistungen und die Entwicklung des Material- und Ausrüstungsbedarfes.

Diese 4265 Grundkennziffern sind für die 15 Bezirke und die DDR zum Teil bis in das Jahr 1960 zurückreichend verfügbar. Der Datenfonds ist für den Nutzer über den Informationskatalog des Verfahrens MWPLNO zugänglich, in dem jede Kennziffer durch folgende Grunddaten identifiziert ist.

- Katalogbezeichnung der Kennziffer,
- Bezeichnung der Kennziffer nach Verfügung DP,
- Maßeinheit,
- Datenquelle,
- Aktualisierungsdaten,
- Plandokument und
- Programm.

Über die Katalogbezeichnung der Kennziffer hat der Nutzer sofort die Möglichkeit, die Kennziffer einem der Sachgebiete Kapazitäten, Grundfonds, Leistungen oder Material- und Ausrüstungsbedarf zuzuordnen. Die Bezeichnung der Kennziffer gemäß einer Verfügung DP ermöglicht ihre Identifikation auf den jeweiligen Arbeitsplatz des Fragestellers. Beide Bezeichnungen können wahlweise bei dem Ausdruck von Ergebnislisten (Querschnittsübersichten, Zeitreihenausdrucke, Vorausberechnungen etc.) genutzt werden. In der Datenquelle wird die genaue Herkunft der Kennziffer aus einem EDVV des Wirtschaftszweiges bzw. aus manueller Datenerfassung einer statistischen Berichterstattung mitgeteilt. Aus den Aktualisie-

rungsdaten ist die Periodizität der Aktualisierung der Kennziffer und die Verfügbarkeit des aktualisierten Datensatzes für den Nutzer ersichtlich. Die Angabe Plandokument weist die Zugehörigkeit einer Kennziffer aus einem Plandokument des Wirtschaftszweiges aus. Über die Information Programm erhält der Nutzer die Informationen, ob für die Kennziffer ein vom Verfahren erarbeitetes Normativ existiert (Normativkatalog), daß er die Möglichkeit hat, für die Kennziffer Zeitreihen bzw. Querschnittsdaten abzufordern und daß er Vorausberechnungen über die Entwicklung der Kennziffer erhalten kann.

2.2. Methodenfonds

Die oben beschriebenen Datenfonds stellen den Ausgangspunkt für eine operative variable Problembearbeitung für unterschiedlichste Problemstellungen aus den vier Sachgebieten Kapazitätsentwicklung, Grundfondsentwicklung, Entwicklung produktiver Leistungen und Material- und Ausrüstungsbedarf dar.

Zur Bearbeitung der Informationen stehen die arithmetischen Operationen zur Verfügung, die es unmittelbar ermöglichen, aus den Grundkennziffern Quotienten, Aggregationen, prozentuale Anteile etc. abzuleiten. Die variable Zusammenstellung von verschiedenen Kennziffern in tabellarischer Form (Matrixkonzept) wird durch die statistischen Verfahren des zur Anwendung kommenden PP STATISTIK gewährleistet. Die Ermittlung von Richtwerten für Planungsgrößen wird durch die Möglichkeiten zur Zeitreihenmodellierung, dem Normativkatalog und Simulationsmodellen auf ökonometrischer Basis unterstützt. Damit enthält die Algorithmenbasis umfassende Möglichkeiten Planungs- und Analyseprobleme zu bearbeiten.

2.3. Technologie

Die Effizienz eines Informationssystems wird wesentlich durch die projektierten technologischen Lösungen zu seiner Handhabung bestimmt. Ein Informationssystem, das über die Befriedigung reiner Auskünfte aus Informationsfonds hinaus geht und das eine variable Problembearbeitung auf der Basis von Datenfonds zum Ziele hat, kann nur in begrenztem Umfange automatisiert werden. Daraus ergibt sich, daß einerseits ein Zwischenglied zwischen Informationssystem und nicht speziell ausgebildetem Nutzer existieren muß

(Anwendergruppe für Informationssystem), andererseits muß eine unmittelbare redundanzarme Reaktionsfähigkeit auf die Nutzerbedürfnisse möglich sein.

Für die praktische Anwendung des mit dem Verfahren MWPLNO bereitgestellten Informationssystems fungiert eine Anwendergruppe als Bindeglied zwischen Nutzer und Informationssystem. Die Verständigungsbasis zwischen Anwendergruppe und Nutzer zur Problembearbeitung bildet der Informationskatalog des Informationssystems. Eine direkt für die selbständige Anwendung durch den Nutzer verfügbare Leistung des Informationssystems ist der Normativkatalog des Verfahrens.

Der Anwendergruppe stehen zur Problembearbeitung eine Reihe von standardisierten Lösungen zur Vorausberechnung und Analyse, das PP STATISTIK/OS mit seiner Algorithmenbasis und für einen Teil der Problembearbeitung die Dialogtechnologie des DSL-Monitorsystems zur Verfügung. Weitere Projektierungsarbeiten konzentrieren sich vor allem auf die direkte Verfügbarmachung der Datenfonds für den Nutzer über Bildschirm (Ankunftssystem) und der weiteren Dialogisierung der Problembearbeitung durch die Anwendergruppe.

Verfasser:
Dr. Wolfram Wittig
ORZ der Deutschen Post
1170 Berlin
Waldpromenade West 4

# DIE ANWENDUNG DER DATENBANKTECHNIK UND DES MENSCH-MASCHINE-DIALOGS IN DER ZENTRALEN STAATLICHEN PLANUNG VON KUBA

Jorge Barrera

Z u s a m m e n f a s s u n g :

Im August 1981 wurde das Einheitliche System der Planbearbeitung (ESPB) im Zentralen Rat für die Planung (Juceplan) der Republik Kuba eingeführt.
Es wird die erste Version dieses Systems und seine Aufgaben und Funktionen beschrieben.

Im August 1981 wurde das Einheitliche System der Planbearbeitung (ESPB) im Zentralen Rat für die Planung (JUCEPLAN) der Republik Kuba eingeführt.
Die erste Version des ESPB ist ein System, das mit Hilfe der Rechentechnik die Erfassung, Speicherung, Verarbeitung und Ausgabe der Daten sichert, die durch die Mitarbeiter des JUCEPLAN für die Ausarbeitung der Volkswirtschaftspläne und für die Kontrolle ihrer Durchführung benutzt werden.
Das ESPB besteht aus einer Menge funktioneller und versorgender Teilsysteme, die die verschiedenen Aufgaben mittels einer koordinierten Arbeitsweise lösen.
Die funktionellen Teilsysteme sind für die Ausführung (oder für die Unterstützung) der Planaufgaben, die im ESPB integriert sind, zuständig. Um ihre Arbeit durchführen zu können, stützen sie sich auf die versorgenden Teilsysteme. Für die erste Version des ESPB wurden 17 funktionelle Teilsysteme entwickelt, die den Planungsabteilungen entsprechen.
Versorgende Teilsysteme sind die folgenden:
- das methodologische Teilsystem,
- das informationelle Teilsystem,
- das mathematische Teilsystem,
- das technische Teilsystem,
- das technologische Teilsystem.

Im Rahmen des vorliegenden Beitrages werden die allgemeinen Mittel der informationellen und der mathematischen versorgenden Teilsysteme vorgestellt.

## 1. Das informationelle versorgende Teilsystem

Die informationelle Versorgung des ESPB hat das Ziel, die Informationen, die für die Ausarbeitung des Volkswirtschaftsplanes benötigt werden, zu untersuchen und die Regeln ihrer Strukturierung, Organisation und Modifikation für einen effektiveren Einsatz der elektronischen Datenverarbeitung zu definieren. Unter informationellen Gesichtspunkten ist der

Volkswirtschaftsplan in letzter Instanz aus Plankennziffern, Klassifikatoren und Nomenklatoren zusammengesetzt.
Eine Plankennziffer ist eine Größe, die das zu erwartende oder reale quantitative Verhalten eines Aspektes des volkswirtschaftlichen Reproduktionsprozesses darstellt. Im ESPB werden die Plankennziffern in primäre und berechnete Kennziffern klassifiziert, wobei die primären Kennziffern die sind, die von den Planern direkt an das System gegeben werden, und die berechneten Kennziffern die sind, die vom System automatisch ausgearbeitet werden.
Nach ihrer Zusammensetzung kann man eine Kennziffer folgendermaßen definieren:

$$\overset{\sim}{KZF} = \{g, n, I\}$$

wobei

g = die quantitative Größe der Kennziffer ist:
m = die Maßeinheit darstellt und
I = $\{i_1, i_2, \ldots, i_m\}$ als die Menge der Elemente (Kodex) definiert wird, die für die Identifizierung der Kennziffer nötig sind. Jedes "i" wird Identifikator genannt.

Als Beispiel einer Plankennziffer kann das Folgende angesehen werden: Die Warenproduktion von Zement im Zweig der Baumaterialienindustrie des Ministeriums für Bauwesen wird im Jahre 1985 5000000 Tonnen betragen (der gegebene Wert soll nur als Beispiel interpretiert werden). Die Komponenten dieser Kennziffer sind folgende:

g = 5000000,
m = Tonnen,
$i_1$ = Warenproduktion,
$i_2$ = Zement,
$i_3$ = Ministerium für Bauwesen,
$i_4$ = Zweige der Baumaterialienindustrie,
$i_5$ = Jahr 1985.

Die Untersuchung der verschiedenen Identifikatoren (i), die für die Darstellung der Plankennziffern benutzt werden, ist einer der Hauptaspekte der informationellen Versorgung des ESPB. Die Identifikatoren sind auf den Domänen (Koordinaten) von Zeit und Raum (geographischer, ökonomischer, organisatorischer usw.) definiert. Ihre Gesamtheit erlaubt die genaue Identifizierung jeder Plankennziffer. Die für die Identifizierung der in der Planung der Volkswirtschaft benutzten Kennziffern der notwendigen Domänen bilden die Klassifikatoren des ESPB.

Die Klassifikatoren werden als kodifizierte Listen der Objektklasse dargestellt. Nach ihrer Zusammensetzung kann ein Klassifikator wie folgt definiert werden:

$$KLA = \{k, A\}$$

wobei:

k = ein Kode ist, der den Schlüssel jedes Elementes des Klassifikators darstellt und

$A = \{a_1, a_2, \ldots, a_n\}$ die Attribute jedes Elementes des Klassifikators sind. Als Attribute können Beschreibungen, Maßeinheiten, verbindliche Kode oder Zahlenwerte vorkommen.

Einer der wichtigsten Klassifikatoren in ESPB ist der Klassifikator für "ökonomische Messungen", der den ökonomischen Inhalt jeder Kennziffer zum Ausdruck bringt. Dieser Klassifikator wird bei jeder Kennziffer benutzt (explizit oder implizit) und definiert den ökonomischen Aspekt des Reproduktionsprozesses, den man messen will (siehe $i_1$ "Warenproduktion" im vorherigen Beispiel). Andere wichtige Klassifikatoren des ESPB sind "ökonomische Aktivitäten" (in denen alle Produkte und Leistungen, die in JECEPLAN geplant werden, enthalten sind), "Wirtschaftsleitende Organe", "Sektoren und Zweige", "Bezirke", "Investitionen" usw.

Ein anderer Begriff, der für die informationelle Versor-

gung des ESPB von Bedeutung ist, ist der Begriff des Nomenklators. Die Nomenklatoren dienen der ausführlichen oder funktionellen Definition der Menge der Kennziffern, die in einem bestimmten Plandokument enthalten sein sollen.
Nach ihrer Zusammensetzung kann ein Nomenklator folgenderweise definiert werden:

$$NOM = \{K, A\},$$

wobei:

$K = \{k_1, k_2, \ldots, k_p\}$ die zugehörigen Kode der Teilmenge der Identifikatoren I ($I = \{i_1, i_2, \ldots, i_m\}$) der Kennziffern, deren Nomenklator zu definieren ist, darstellen.

Wenn $m = p$, dann wird die Nomenklatur in ausführlicher Weise definiert, und wenn $m > p$, wird sie in funktioneller Weise definiert.

Weiterhin spricht man von einem einfachen Nomenklator, wenn $p = 1$, und wenn $p > 1$, sagt man, daß der Nomenklator sei zusammengesetzt.

$A = \{a_1, a_2, \ldots, a_n\}$ stellt wie bei den Klassifikatoren eine Menge von Attributen dar.

Um die Größe, Kompliziertheit und Variabilität der Struktur der Datenbasis von JUCEPLAN beherrschen zu können, wurden für die Datenbank des ESPB drei vereinfachte Datenmodelle benutzt, welche die vorher definierten Datenarten widerspiegeln können und als Klassifikatoren, aktualisierbare Belege und Ergebnistabellen bezeichnet werden.
Diese Datenmodelle wurden ausgehend von den Erfahrungen des Hauptrechenzentrums des GOSPLAN der UdSSR und der Arbeiten von E. F. Codt über das relationale Datenmodell entwickelt /5/6/9/.
Das Datenmodell der Klassifikatoren schließt die Klassifikatoren und die Nomenklatoren des ESPB ein; sie werden als Liste von Relationen (oder n-tupeln) dargestellt, wo jede Domäne entweder ein Kode oder ein Attribut sein kann. Um

tiefer in die Begriffe Relation, Domäne und n-tupel einzudringen, kann man die bibliografischen Referenzen /5/, /7/, /11/ und /12/ benutzen.
Das Datenmodell der aktualisierbaren Belege entspricht den primären Kennziffern, auch wenn eventuell einige berechnete Kennziffern, die während des Erfassungsprozesses erhalten werden, hier auch eingeschlossen sind.
Dieses Datenmodell sowie das der Ergebnistabellen hat das Ziel, den Benutzer (Planer) eine Darstellung der Daten anzubieten, die den normalen Plandokumenten, die seit jeher benutzt werden, so ähnlich wie möglich ist. Diese Darstellung beruht auf der bekannten tabellarischen Einordnung der Daten in Spalten und Zeilen (siehe die Ähnlichkeit mit dem Relationsbegriff).
Bei den automatisierbaren Belegen wird die Information in 3 Ebenen eingeteilt: Beleg, Seite, Zeile.
Die Belege sind übereinstimmende Gruppierungen von Plankennziffern, die eine einheitliche oder kompatibele Struktur haben.
Die Seiten sind Teilmengen von Belegen, die nicht mehr als 20 Zeilen und eine vom Beleg abhängige Anzahl von Spalten umfassen. In einer Seite können die Kennziffern eines oder mehrerer gleichwertiger Identifikatoren enthalten sein. Diese Identifikatoren werden im oberen Teil der Seite (Kopf der Seite) gesetzt und logische Identifikatoren des Kopfes (LIK) genannt, während der Rest der Identifikatoren, die normalerweise für jede Kennziffer verschieden und deswegen in jeder Zeile zu schreiben sind, als logische Identifikatoren der Zeile (LIZ) bezeichnet werden. Die Zeilen sind Mengen von Identifikatoren und quantitativen Werten, wobei jeder einer Spalte entspricht, die im Datenmodell in horizontaler Weise eingesetzt werden.
Das ESPB sichert dem Benutzer den Zugriff zu einer Seite eines gegebenen Beleges im Dialogbetrieb und über die Stapelverarbeitung den Zugriff zu einer Gruppe von Seiten oder

zum ganzen Beleg.
Das Datenmodell der Ergebnistabellen entspricht den Ergebnisformationen des ESPB, die aus primären und berechneten Kennziffern zusammengesetzt sind. Ihre Struktur ist der der Belege ähnlich, mit dem einzigen Unterschied, daß es keine Grenze für die Anzahl der Zeilen, die zu einer Seite gehören, gibt.
Wie bei den Belegen existieren auch im Datenmodell der Ergebnistabellen die Begriffe "logischer Identifikator des Kopfes" und "logischer Identifikator der Zeilen" (LIZ).
Das ESPB sichert dem Benutzer den Zugriff zu der ganzen Ergebnistabelle, zu den Zeilen, die den gleichen LIK haben (Seite bei den Ergebnistabellen) und zu jeder einzelnen Zeile.

2. Das mathematisch versorgende Teilsystem

Das mathematisch versorgende Teilsystem ist dazu bestimmt, die nötigen Programmittel für die Lösung der Aufgaben des ESPB in den EDV-Anlagen auszuarbeiten und weiterzuentwikkeln.
Um die rechentechnischen Funktionen der Erfassung, Berechnung und Ausgabe sowohl im Stapelbetrieb als auch in einem Mensch-Maschine-Dialog bei den unterschiedlichen Merkmalen jedes der funktionellen Teilsysteme zu sichern, war es nötig, bei den Elementen der Datenfonds sowie bei den rechentechnischen Verarbeitungsverfahren, im Rahmen der informationellen bzw. mathematischen versorgenden Teilsysteme eine Menge von Gestaltungsmaßnahmen zu treffen. Woraus eine allgemeine standardisierte Gestaltung der Hauptelemente beider Teilsysteme erzielt wurde.
Diese allgemeine standardisierte Gestaltung wird in jedem einzelnen Fall durch Parameter präzisiert, was die rationelle Beherrschung der großen Variabilität der Struktur der

Daten und der Verarbeitungsverfahren ermöglicht. Die Parameter bestimmen demzufolge sowohl die präzise Lage der Daten als auch die Einzelheiten der auf sie anzuwendenden Verarbeitungsverfahren. Die allgemeine mathematische Versorgung des ESPB ist aus einer Menge von Algorithmen, Programmen und Hinweisen zusammengesetzt, die mittels eines Parameterkomplexes den Hauptanteil (etwa 90 %) der rechentechnischen Funktionen der Erfassung, Berechnung und Ausgabe jedes funktionellen Teilsystems und insbesondere alle die Verarbeitungen, die direkt mit den gespeicherten Daten zu tun haben, durchführt.
Für die technische Realisierung des ESPB wurden außer der Grobbeschreibung der Nutzerfreundlichkeit der allgemeinen mathematischen Versorgung für die verschiedenen Benutzer (Planer, Organisatoren, Programmierer und EDV-Betriebsspezialisten) folgende Anforderungen an diese Mittel gestellt:

- Existenz eines einzigen gespeicherten Datenfonds (als Datenbank), der gleichzeitig in einem Mensch-Maschine-Dialog und im Stapelbetrieb betrieben werden kann und der die notwendige Sicherheits- und Schutzmaßnahmen besitzt;
- Modulare Gestaltung der Programmkomplexe mit dem Ziel, eine progressive Realisierung und Inbetriebnahme sowie nachfolgende Wartungen zu erleichtern;
- Optimale Besetzung der verfügbaren technischen Mittel.

Ausgehend von diesen Anforderungen und unter Beachtung sowohl der Ergebnisse der früheren Untersuchungen und Forschungen sowie der Charakteristiken der staatlichen zentralen Planung als auch der verfügbaren technischen Versorgung wurde die allgemeine mathematische Versorgung des ESPB in die folgenden Programmkomplexe eingeteilt:

- Datenbankbetriebssystem (DBBS),
- Wiederherstellungsprozedur,
- Kohärenzkontrolle,

- Wartung der Dateien;
- Parametererfassung,
- Stapelerfassung;
- Zurückgewinnung,
- Laden der Tabellen,
- Ausladen der Tabellen,
- Tabellengenerator,
- System für den Dialog durch die Bildschirmgeräte (SDB),
- Einflußdirektor.

Das Hauptmerkmal der gewählten allgemeinen Gestaltung ist das Betrachten aller dieser Programmkomplexe als sequentielles zusammenlaufendes System /2/, /4/, da sie während ihrer Laufzeit in verschiedenen Bereichen des Speichers untergebracht werden. Die Kommunikation zwischen all diesen Programmkomplexen wird bei der Anwendung der Technik des "Postfaches", die bei dem Betriebssystem SIRIS-III der IRA-50 vorhanden ist, erreicht. /1/+

Das Prinzip dieser Technik beruht darauf, daß ein Programm eine oder mehrere Datenzonen (Postfächer) erklären kann, wo es Nachrichten von anderen Programmen, die in anderen Speicherregionen arbeiten, erhalten kann. Die erhaltenen Nachrichten bilden eine Warteschlange und werden nach den Bedürfnissen des betreffenden Programms abgearbeitet. Jedes im Betrieb stehende Programm kann Nachrichten zu den Postfächern, die von anderen Regionen erklärt wurden, schicken.
Die Existenz eines einzigen Programmkomplexes für das Betreiben der Datenbank sowie die Anwendung der Postfächertechnik sichern die Erlangung der Anforderungen über den

+ Dem Verfasser ist keine andere Anwendung der Postfachtechnik als Grundlage der Bestimmung der Architektur eines Datenbankbetriebssystems bekannt, obwohl die Ergebnisse gezeigt haben, daß diese Technik gerade für die Datenbankproblematik sehr geeignet ist.

gleichzeitigen Stapel- und Dialogbetrieb der Datenbank und über die modulare Gestaltung der Programmkomplexe, die an die allgemeine mathematische Versorgung des ESPB gestellt wurden.
Außer den vorgenannten Programmkomplexen, die zu allgemeinen mathematischen Versorgung des ESPB gehören, schließt dieses Teilsystem auch folgende typischen mathematischen Versorgungsmittel ein:

- Programmkomplexe für die Zusammenfassung und Berechnung von Kennziffern im Stapelbetrieb;
- Routine für die Zusammenfassung und Berechnung der Echtzeit.

Im weiteren werden die Hauptmerkmale dieser Programmkomplexe beschrieben.
Das Datenbankbetriebssystem (DBBS) ist ein Programmkomplex, der die Datenübertragung zwischen dem Dateikomplex, in dem physisch die Daten der Datenbank des ESPB gespeichert sind, und den restlichen Programmkomplexen durchführt. Die restlichen Programmkomplexe werden auf diese Weise von der Notwendigkeit, die physische Struktur der gespeicherten Daten zu kennen, befreit.

Die Gesamtheit der Daten des ESPB wird in drei Hauptdateien eingeteilt, zwei von ihnen haben eine undefinierte Organisation und eine hat eine sequentielle Organisation. Aus diesen Dateien kann jedes Programm die elementare Dateneinheit:

- Das Attribut bei den Klassifikatoren,
- Die Seite bei den aktualisierbaren Belagen,
- Die Zeile bei den Ergebnistabellen

sogar in direktem Zugriff, als auch sequentiell nach verschiedenen Kriterien, unter Benutzung der Postfachtechnik durch das DBBS zurückgewinnen oder schreiben.
Dafür wurde eine Datenmanipulationssprache (DMS) entwickelt,

welche aus etwa 60 Anweisungen zusammengesetzt ist und die von jedem Programmkomplex für die Zurückgewinnung und das Schreiben der Daten aus bzw. in der Datenbank über die Postfächer benutzt wird. Weiterhin existiert eine COBOL-Schnittstelle, die es erlaubt, die DMS auch in den in dieser Sprache geschrieben, nicht allgemeinen Programmen, zu benutzen.

Die Speicherung der Klassifikatoren und Nomenklatoren wurde in einer nichttraditionellen Weise gestaltet. In der Datenverarbeitungspraxis ist es üblich, alle die Attribute, die zu einem Objekt innerhalb einer Objektklasse gehören, in einem physischen Satz zusammenzustellen. Diese physische Organisation der Klassifikatoren ist aber nur effektiv für den direkten Zugriff aller Attribute eines Objekts, was besonders in der Aktualisierung der Klassifikatoren gebraucht wird, und für die sequentielle völlige Zurückgewinnung aller Attribute einer Gruppe von Objekten innerhalb einer Objektklasse, was für die Listen der Klassifikatoren oder für andere, im allgemeinen nicht von den Kennziffern abhängige Gesamtverarbeitungen aller Attribute einer Gruppe von Objekten ist.

Die Klassifikatoren und Nomenklatoren haben aber als eines ihrer Merkmale, daß sie eine große Stabilität aufweisen, und daß deswegen ihre Aktualisierungsrate in der Praxis vernachlässigt werden kann. Das Listen der Klassifikatoren oder ihre gesamte Verarbeitung wird auch sehr selten gemacht. Die häufigste Benutzung der Klassifikatoren findet während der Ausarbeitung der Ergebnistabellen statt. Dieser Prozeß kann aber so organisiert werden (programmäßig), daß die Zurückgewinnung der notwendigen Attribute nicht für jede Zeile, sondern auf einmal für eine Gruppe von Zeilen (zum Beispiel für eine Seite) durchgeführt wird.

Die gewählte physische Organisation der Klassifikatoren, die das Ziel verfolgt, diese letzte Art der Arbeit mit

den Klassifikatoren zu optimieren, setzt als physische Speicherungseinheit die Liste aller Werte jedes Attributes, d. h. alle die Realisierungen eines und desselben Attributes fest. In der Sprache des relationellen Datenmodells könnte man es so ausdrücken, daß die ganze Domäne die Speicherungseinheit darstellt, d. h. daß die Relationen nicht nach Tupeln, sondern nach Domänen gespeichert werden.
Die Beziehungen zwischen den Werten verschiedener Attribute, die zu einem und demselben Objekt gehören, werden durch eine "Verwirklichungsnummer gesichert", die der ordinalen Stelle jedes Wertes in der zugehörigen Liste entspricht. Für alle zum selben Objekt gehörenden Attributwerte ist sie dieselbe. Die Verwirklichungsnummer der Objekte eines Klassifikators werden nach der im System chronologischen Eintrittsfolge ausgewiesen. Das Komplement dieser in undefinierter Weise organisierten Datei ist eine Direktzugriffsdatei, in der ein Satz für jeden Schlüssel eines Klassifikators oder Nomenklators existiert und die die zugehörige Verwirklichungsnummer enthält. Das ermöglicht den Zugriff zur vorher erklärten undefinierten Datei, um den Wert eines Attributes beim Vorhandensein des Wertes des zugehörigen Schlüssels zu kennen. Die praktische Anwendung dieser Speicherungsform hat bewiesen, daß sich die Anzahl der physischen Zugriffe zu den Dateien für die Vorbereitung einer Tabelle bis um das 30-fache im Vergleich mit der traditionellen Methode vermindert.
Die physische Speicherungsform der aktualisierbaren Belege im Datenfonds wurde unter Berücksichtigung der Antwortgeschwindigkeit, der Flüssigkeit der Verarbeitungsprozesse und der Sicherheit der Daten entschieden. Die physische Speicherungseinheit ist die Seite eines Beleges in einer Datei mit undefinierter Organisationsform. Die Eingänge und Ausgänge in den bzw. aus dem Datenfonds werden demzu-

folge immer über ganze Seiten durchgeführt. Die Adresse jeder Seite wird dabei in einer Datei mit direktem Zugriff eingeschrieben. Für einen Beleg kann es bis 20 Sätze dieser Datei geben, deren Schlüssel die Belegnummer und der Satznummer (von 1 bis 20) ist. Jeder Satz hat 250 Adreßverweise, die die Lage des ersten Bytes der Seite in der undefinierten Datei enthält. Mit dieser Organisation kann man eine Seite in folgender Weise zurückgewinnen, wenn man ihre Nummer kennt.

- Man errechnet die Satznummer der Datei, die die Adreßverweise enthält, durch die Formel:

$$\text{Satznummer} = \frac{\text{Seitennummer}}{250} + 1$$

- Man sucht in der Datei mit direktem Zugriff den Satz mit dem Schlüssel Belegnummer - Satznummer
- Mit dem Rest der Division Seitennummer/250 bekommt man im vorher gelesenen Satz die Adresse der Seite.

Darüber hinaus gibt es bei den aktualisierbaren Belegen eine dritte Datei mit direktem Zugriff, die den Schlüssel der Belegnummer "logischer Identifikator des Kopfes" (LIK) hat. Sie enthält die verschiedenen Seitennummern, die zum gegebenen LIK gehören, so daß auch der Zugriff zu einer bestimmten Information möglich ist, wenn ihre physische Seitennummer nicht bekannt ist.
Bei den Ergebnistabellen ist die physische Speicherungseinheit die Zeile einer Tabelle. Alle Zeilen aller Tabellen werden in einer index-sequentiellen Datei gespeichert. Um die Redundanz des Kodes zu vermindern und etwas Platz zu gewinnen, gibt es im Dateikomplex eine zweite Datei für die Ergebnistabellen, die den Schlüssel Tabellennummer - LIK hat, und die eine laufende Nummer für jeden Satz besitzt. Auf diese Weise wird der Schlüssel der ersten Datei folgendermaßen zusammengesetzt:

- laufende Nummer (von zweiter Datei angegeben),

- logischer Identifikator der Zeile (LIZ).

Der Zugriff zu den in der Datenbank gespeicherten Informationen wird für die Mitarbeiter des JUCEPLAN, die den Zustand von "Abonenten" erhalten haben, begrenzt. Das Betreiben des Systems durch die Bildschirme fördert die Durchführung von automatischen Schutzmaßnahmen, die bei dem ESPB in zwei Ebenen gegliedert sind:

- die Überprüfung des Zustandes des Benutzers als Abonent des Systems am Anfang einer Sitzung,
- die Überprüfung des Rechtes des Abonenten, zu einer bestimmten Information in der Datenbank zuzugreifen.

Beide Schutzmaßnahmen werden vom DBBS mit Hilfe einer indexsequentiell organisierten Datei, die Abonentendirektorium genannt wird, automatisch durchgeführt.
Das DBBS sichert weiterhin die Datenbank vor Unfällen oder Operationsfehlern, die die gespeicherten Informationen in unerwünschter Weise verändern oder zerstören. Das erreicht man mit Hilfe einer auf Magnetband gespeicherten Datei, die alle die Datenbank verändernden Anrufe zum DBBS enthält, und die laufend nach der Durchführung der zugehörigen Operationen von DBBS geschrieben wird.
Die Wiederherstellungsprozedur ist dafür bestimmt, im Falle des Vorkommens irgend eines Umstandes, der die Datenbank zerstört, den Inhalt desselben bis zu dem der Zerstörung vorhergehenden Moment wiederherzustellen.
Alle Anweisungsfolgen an das DBBS bei Benutzung der DMS benötigen eine Eröffnungsanweisung vor ihrer Durchführung und eine Endanweisung nach der vollen richtigen Beendigung des Verarbeitungsschrittes. Diese Eröffnungs- und Endanweisungen werden auch in die Sicherheitsdatei geschrieben. Die Wiederherstellungsprozedur analysiert die Sicherheitsdatei des DBBS und befiehlt dem System die Anweisungen zu

wiederholen, die den Inhalt der Datenbank geändert haben und die einwandfrei beendet wurden. Dieser Prozeß vollzieht sich natürlich auf der Grundlage der Kopie der Datenbank eigenen Datei durchgeführten Wiederherstellung der Anfangsbedingungen.
Auf diese Weise wird gesichert, daß im Falle der Zerstörung der Datenbank maximal die Aktualisierungen der in diesem Moment in den Bildschirmen stehenden Seiten verlorengehen.
Die Kohärenzkontrolle hat als Ziel, die automatische Überprüfung der Richtigkeit der Adreßverkettungen und sonstigen Serviceinformationen zu ermöglichen, die zwischen den bzw. in den die Datenbank bildenden physischen Dateien vorhanden sind.
Weiterhin werden dabei die notwendigen Informationen gegeben, um im Falle der Entstehung irgend eines Fehlers denselben analysieren zu können. Bei der Benutzung der Datenbankbetriebssysteme treten oftmals Inkohärenzen in oder zwischen den physischen Dateien auf, die zum DBBS gehören. Diese Tatsache hat als Ursache, daß die Durchführung einer Schreibanweisung, die zum DBBS gegeben wird, normalerweise in mehrere physische Schreibbefehle übersetzt wird, die alle vollständig durchgeführt werden müssen, da es, wenn es nicht so geschieht, ohne weiteres vorkommen kann, daß die Zugriffsalgorithmen nicht mehr richtig arbeiten.
Ein einfaches Beispiel für diese Tatsache findet man bei den Dateien mit index-sequentieller Organisation, bei denen, wenn ein neuer Satz hinzugefügt wird, es nötig ist, den Satz in der Datei zu schreiben und die Indextabellen und Adreßverkettungen zu den anderen Sätzen zu pflegen. Wenn es irgendeine Unterbrechung der Arbeit der EDV-Anlage (Ausfall der Energie, Zusammenbruch des Operationssystems usw.) während der Zeitspanne, die für die Durchführung der oben genannten physischen Schreibbefehle vor-

kommt, ist es selbstverständlich, daß die Datei in einem unbenutzbaren Zustand bleibt.
Da bei den DBBS noch komplexere Verkettungen in oder zwischen den Dateien nötig sind, nimmt die für die völlige Durchführung einer den DBBS gegebenen Schreibanweisung nötige Zeitspanne und damit auch die Wahrscheinlichkeit einer Zerstörung der Datenbank bei einer Arbeitsunterbrechung der EDV-Anlage zu.
Die Wartung der Dateien ist zu machen, wenn die unter der Kontrolle des vom Hersteller der EDV-Anlage gelieferten Dateibetriebssysteme befindlichen physischen Dateien, wegen ähnlicher Ursachen wie die vorher erklärten, zerstört werden.
In diesem Fall wird oftmals die Unregelmäßigkeit nicht gleich durch die Kohärenzkontrolle entdeckt, sondern in einem Moment, in dem die Wiederherstellung der Datenbank durch die übliche vorher erklärte Prozedur sehr aufwendig wäre.
Bei der Ausnutzung der in der Datenbank vorhandenen Redundanz ist der Programmkomplex für die Wartung der Dateien in der Lage, alle Verzeichnisdateien der Klassifikatoren und der aktualisierbaren Belege völlig neu zu schreiben, die einer solchen Zerstörung ausgesetzt waren.
Da sie eine Direktzugriffsorganisation haben, läßt sich von den Dateien ausgehen, die die Informationen als solche haben, die Dank ihrer undefinierten Organisation nicht vom Dateibetriebssystem zerstört werden können.
Die anderen Dateien des DBBS, die eine index-sequentielle Organisation haben, werden mit Hilfe von Nutzerhilfsprogrammen, die vom Hersteller der EDV-Anlage geliefert werden, gewartet.
<u>Die Parametererfassung</u> erfüllt folgende Aufgaben:

- Syntaktische Kontrolle aller Parameter, die sowohl für die Beschreibung der Geographie der verschiedenen Da-

tenarten als auch für die Bestimmung der auf sie anzuwendenden standardisierten Prozeduren in die Datenbank eingeführt werden.

- Die Übersetzung des externen (Quell-) Formates der Parameter auf ein internes (Objekt-) Format, das für die Vereinfachung des Betriebes der Programme gestaltet wurde, die die mathematische allgemeine Versorgung des ESPB bilden.
- Ausgabe von Listen über die in der Datenbank geladenen Quellparameter.

Die Stapelerfassung der aktualisierbaren Belege wird mittels einer Programmkette, die folgende Funktionen umfaßt, durchgeführt:

- physische Kontrolle der Eingangsdaten,
- Aktualisierung der Belege,
- logische Kontrolle der aktualisierten Daten,
- Laden der aktualisierten Belege in der Datenbank,
- Listen des Abbildungsprotokolls der in der Datenbank geladenen Daten.

Die Aktualisierung der Belege wird mittels der klassischen Funktionen Einfügen, Verändern, Löschen durchgeführt, wobei das Einfügen immer nach Zeilen, das Verändern nach Zeilen oder Zeilen/Spalten und das Löschen nach Zeilen oder Spalten durchgeführt werden kann.
Die Einteilung in physische und logische Kontrollen der Eingangsdaten entspricht der Auffassung, daß das Rechenzentrum nur dafür verantwortlich ist, die exakte Übertragung der auf Belegen enthaltenen Daten auf maschinenlesbare Datenträger durchzuführen, was durch die physische Kontrolle gesichert wird, während die Benutzer (Planer) für den Inhalt der Belege verantwortlich sind, deren syntaktische Fehler durch die logischen Kontrollen gezeigt

werden. /3/
Außer den vorgenannten Funktionen sichert dieser Programmkomplex, daß die Zeilen, die zu einem logischen Identifikator des Kopfes gehören, immer rechtmäßig innerhalb der Seiten, zu denen sie gehören, sortiert bleiben.
Die Listen des Abbildungsprotokolls der in der Datenbank geladenen Daten können wahlweise nur für die Seiten, die modifiziert waren, oder für den ganzen Beleg gedruckt werden.
Dieser Programmkomplex wird auch für das Anfangsladen eines Beleges, der nicht auf Papier, sondern auf irgend einem anderen maschinenlesbaren Datenträger kommt sowie für die Ermittlung der Abbildungsprotokolle der Seiten, die mit Hilfe der Bildschirme in einem Mensch-Maschine-Dialog modifiziert waren, benutzt. Die letztgenannte Funktion wird durch die Verarbeitung der Sicherheitsdatei erreicht.

Die Stapelerfassung der Klassifikatoren umfaßt folgende Funktionen:

- Kontrolle der Eingangsdaten,
- Aktualisierung der zugehörigen Dateien,
- Ausdruck der durchgeführten Aktualisierungen,
- Listen der Klassifikatoren.

Die Zurückgewinnung der Daten, die in der Datenbank gespeichert sind, ermöglicht, relationale Dateien zu bekommen, ausgehend von den Belegen, den Klassifikatoren und Nomenklatoren.
Bei der ersten Version des ESPB wurde dieser Programmkomplex in einer minimalen Version eingeführt, die nur die natürliche Vereinigung /6/ zwischen einem Beleg und den Klassifikatoren und Nomenklatoren löst. Für zukünftige Versionen ist vorgesehen, diesem Programmkomplex schrittweise soweit zu erweitern, daß er den Umfang der relationalen Algebra erreicht.

Das Laden von Ergebnistabellen hat nun die Funktionen der Vereinigung aller berechneten Dateien, die alle Tabelle geladen werden sollen, ihre Umformung zum Datenbankladeformat und das eigentliche Laden in die Datenbank.

Das Ausladen von Ergebnistabellen führt zur umgekehrten Umformung, d.h. vom Datenbankformat zum Format der berechneten Dateien und schreibt die entsprechenden Dateien auf Magnetbänder.

Der Tabellbhgenerator ist ein Programmkomplex, der die Aufgabe hat, alle die Fragen, die mit der Einsetzung der Daten auf den Seiten und mit der Einfügung der konstanten Texte und der Attribute zu tun haben, zu lösen, um, ausgehend von den berechneten Dateien, Ergebnistabellen mit der notwendigen Qualität auszudrücken.

Das System für den Dialog durch die Bildschirmgeräte (SDB) des ESPS erlaubt den Planern, direkt mit der Datenbank zu arbeiten, durch die Erfüllung folgender Funktionen:

- Aktualisierung der Klassifikatoren,
- Aktualisierung der Belege,
- Ansehen der in der Datenbank gespeicherten Ergebnistabellen.

Andererseits erlaubt auch dieser Programmkomplex dem Planer, seinen persönlichen Schlüssel (Abgangschlüssel) zu ändern, wenn er voraussetzt, daß er jemandem anderen bekannt ist.

Die Hauptmerkmale dieses Programmkomplexes sind folgende:

- Anwendung der Autodatentheorie für die Gestaltung der verschiedenen Programmodule,/13/
- Anwendung von Frage-Antwortprozeduren (geschlossener Dialog)*

* siehe Seite 9797

mit der Signalisierung der möglichen Antworten als Grundgedanken für die Gestaltung des Dialoges;

- Möglichkeit der Auswahl zwischen verschiedenen Versionen des Dialoges mit dem Ziel, die durchzuführenden Anschläge für eine bestimmte Arbeit minimieren zu können;
- Anwendung der Wiederbenutzungstechnik (gleichzeitige Benutzung ein und desselben Programme für mehrere Aufgaben) für die Realisierung der Programmodulen;
- Gestaltung und Einführung einer besonderen Prozedur für die dynamische Nutzung des Operationsspeichers, mit sekundären Speicherungen auf Magnetplatten mit dem Ziel, den notwendigen Operationsspeicher zu minimieren, bei der Benutzung des Umfangsunterschiedes zwischen den Reaktionszeiten des Menschen und der Maschine (nach den Untersuchungen in JUCEPLAN liegt dieser Unterschied in der Größe 10 : 1).

Die maximale Reaktionszeit des Dialoges bei den elementaren Funktionen übersteigt nicht 4 Sekunden, was nach den psychologischen Forschungen eine optimale Zeit ist (siehe

---

+ von Seite 96

Im allgemeinen gibt es zwei Möglichkeiten, um einen Mensch-Maschine-Dialog zu gestalten:

- offener Dialog, wo der Benutzer seine Wünsche in Form von Programmen (auch wenn sie sehr einfach sind) formuliert. Bei dieser Art von Dialogen unterscheidet man weiter zwischen prozeduralen und nichtprozeduralen Sprachen (die relationale Algebra bzw. das relationale Kalkül sind Beispiele von Sprachen für die Zurückgewinnung von Daten, die offen funktionieren können.
- geschlossener Dialog, wo der Benutzer an Hand von festgelegten Frage-Antwort-Prozeduren und den zugehörigen Menue ihre Wünsche allmählich präzisiert, bis sie von der EDV-Anlage durchgeführt werden können.

für Fachkader, die nicht in der EDV spezialisiert sind, ist die zweite Variante vorteilhafter, da mit ihr die Ansprüche an den EDV-Kenntnissen der Kader sehr niedrig gehalten werden können und somit das psychologische Hindernis, das immer bei der Anwendung der EDV zu finden ist, leicht bewältigt werden kann.

/8/ und /10/. Außer den vorhergenannten Funktionen sendet, wenn es gewünscht wird, das SDB zum Einflußdirektor den ursprünglichen Wert und den neuen Wert jeder primären Plankennziffer, die modifiziert wird mit dem Ziel, die zugehörigen, datenbankgeladenen Ergebnistabellen im Echtzeitbetrieb zu aktualisieren.

Der Einflußdirektor ist eine Schnittstelle zwischen dem SDB und dem DBBS einerseits und dem Routinekomplex andererseits, der die Aktualisierung der in der Datenbank geladenen Ergebnistabellen im Echtzeitbetrieb sichert, ausgehend von den Modifikationen der primären Plankennziffern, die durch die Bildschirme durchgeführt wurden.
Im weiteren werden die Hauptmerkmale der typischen mathematischen versorgenden Mittel beschrieben, die in der ersten Version des ESPB eingeschlossen sind.

Die Stapelberechnung und -aggregierung der Plankennziffern kann man im ESPB durch einen parametrisierten Programmkomplex, der folgende Funktionen sichert, durchführen:

- arithmetische Operationen zwischen den zu dem selben Satz gehörenden Daten;
- arithmetische Operationen zwischen zu verschiedenen Sätzen gehörenden Daten;
- Bildung neuer Sätze durch die Aggregierung oder andere arithmetische Berechnungen zwischen den Daten, die zu den zu verarbeitenden Sätzen gehören.

Bei diesem Programmkomplex haben sowohl die Ergebnisdateien als auch die Eingangsdateien eine Geographie mit denselben Prinzipien, die für die zurückgewonnenen Dateien benutzt werden. Auf diese Weise wird eine große Flexibilität für die Gestaltung der Berechnungen, für die Benutzung des Tabellengenerators oder für das Laden und Ausladen der Ergebnistabellen in der Datenbank er-

reicht.
Wenn die Möglichkeiten, die von diesen Mitteln geboten werden, nicht für die Lösung einer bestimmten Aufgabe ausreichen, oder wenn aus Effektivitätsgründen die Benutzung eines anderen Algorithmus empfohlen wird, kann man die Umwandlung der zurückgewonnenen Dateien in berechneten Dateien mittels spezifischer mathematischer Versorgungsmittel durchführen.
Die Berechnung und Aggregierung in Echtzeit der Ergebnistabellen wird, wie vorher gesagt wurde, mittels eines Routinekomplexes erreicht, der vom System für den Dialog durch den Einflußdirektor den ursprünglichen und neuen Wert jeder primären modifizierten Plankennziffer bekommt. In Abhängigkeit vom aktualisierten Beleg ruft der Einflußdirektor die zugehörige Routine auf, die den "Einfluß" der Veränderung der modifizierten primären Kennziffer auf die in den Ergebnistabellen gespeicherten Kennziffern, die damit zu tun haben, berechnet.
Diese Prozedur sichert, daß sich die aktualisierbaren Belege und die Ergebnistabellen auf Dauer auf demselben Aktualisierungsniveau in der Datenbank stehen, was einerseits die Qualität bestimmter Planungsaufgaben um ein Vielfaches erhöht und andererseits die EDV-Anlagenzeit für die tägliche Stapelberechnung der Tabellen spart.
Für die detaillierte Gestaltung und die Realisierung der allgemeinen mathematischen Versorgung wurden 480 Mathematiker-Monate verwendet. Die Anzahl der geschriebenen und Betrieb gesetzten Anweisungen beläuft sich auf 79076 Befehle im IRIS-assembler (ASSIRIS) und auf 20825 Anweisungen im COBOL.

Literaturquellen

/1/ Monitor SIRIS-III Anwendungshandbuch
4002 EY/FR (franz.), Paris 1976

/2/ Alaban, u. a.
Der Zusammenlauf von Systemen auf Mikrorechner
Revista Iberolem erizana de computacion Buo II No 14
Madrid 1981 (span.)

/3/ Barrera, J. u. a.
Allgemeines System für die Erfassung und Speicherung von Daten (SGCAD)
Vortrag zum Forum XV. Jahrestag von IMACC (span.)
Akad. der Wissenschaften Kubas, La Havana 1979

/4/ Bernstein
Zusammenlaufkontrolle in geteilten Datenbanksystemen
Comptuting Serrveys ACM vol 13. No 2 Juni 1981 (engl.)

/5/ Codd, E. F.
Ein relationales Modell für große verteilte Datenbanken
Comm. of the ACM No 6. 1970 (engl.)

/6/ Janimiev, B. T.
Kodierung der ökonomischen Informationen unter den Bedingungen des ASPR (russ.)
Estatistika, Moskwa 1976

/7/ Maininas, E. S.
Planungsprozesse, Informationsaspekt
Verlag Die Wirtschaft Berlin 1972

/8/ Mertens, P.
Die technische Gestaltung von Bildschirmdialogen
online - adl-Nachrichten 7/77, S. 670 - 675

/9/ Proskurov, O.
Die Information im ASPR
Ekonomika, Moskau 1975 (russ.)

/10/ Rohlfs, S.
Antwortzeiten im Dialog
Online - adl-Nachrichten 9/77, S. 676 - 677

/11/ Senko, M. E.
Datenstrukturen und Zugriffe zu den Datenbasesystemen: Vergangenheit, Gegenwart und Zukunft
L'Informatique Nouvelle No. 106, Paris 1979 (franz.)

/12/ Wedekind, H.
Datenbanksysteme I.
Informatik Bd. 16
Wissenschaftsverlag Meisenheim/ Glan 1974

/13/ Urinson, Ja.
Erhöhung der Effektivität der Anwendung ökonomisch-mathematischer Modelle im ASPR
Planovoe chozjaistvoe, 6, 1978, S. 59 - 66

Dipl-Ing. ök. Jorge Barrera
Direktor des Rechenzentrums von
Juceplan, Havana, Republik Kuba

Pester, Matthias —

## Zur Parallelisierung von Datenverwaltung und -verarbeitung in Programmsystemen

### 1. Einleitung

Wissenschaftlich-technische Aufgabenstellungen, bei denen die zur formalen Beschreibung des konkreten Problems erforderlichen Daten einen solchen Umfang annehmen, daß sie im Hauptspeicher einer EDVA nicht gleichzeitig darstellbar sind, bilden aufgrund des hohen Leistungsvermögens von Informationsverarbeitungssystemen heute keine Seltenheit mehr. Die im Verlaufe des Lösungsprozesses solcher Aufgabenklassen zu verarbeitenden und neu anfallenden Daten werden in einer Speicherhierarchie dargestellt, deren bekanntester Vertreter der virtuelle Speicher (VS) ist /7/. Während jedoch die Verwaltung dieser zweistufigen Speicherhierarchie durch die VS-Steuerung problemunabhängig erfolgt, kann eine problemangepaßte Organisation der Speicherhierarchie bei genauer Analyse von Problem- und Datenstrukturen zu einer effektiveren Gesamtlösung führen, wenngleich sich der Entwicklungsaufwand i. a. erhöht.
Eine solche Aufgabenklasse ist beispielsweise die Behandlung des allgemeinen Eigenwertproblems

$$(A - \lambda B)X = 0$$

für großdimensionierte schwachbesetzte symmetrische Matrizen A und B, wofür an der Sektion Mathematik der TH Karl-Marx-Stadt ein Programmpaket entwickelt wurde /3/. Diese Aufgabenstellung ergibt sich u. a. aus der elastostatischen Berechnung von Bauteilen mit Hilfe der Methode der finiten Elemente.
In diesem Programmpaket ist eine problemorientierte Verwaltung der Speicherhierarchie enthalten, der das Konzept der abstrakten Datentypen zugrundeliegt, wobei eine möglichst effektive Datenverwaltung durch Minimierung der Anzahl der Transferoperationen und weitgehende Parallelisierung von Arithmetik und Transfer angestrebt wurde.

## 2. Problemorientierte Datenverwaltung

Unter einer Datenverwaltung werden alle Funktionen und Prozesse verstanden, die unmittelbar verantwortlich sind für die Vergabe von Zugriffsrechten zu ausgewählten Daten an die Programmoduln. Dabei wächst der Anteil der Datenverwaltung am Gesamtlösungsaufwand sehr rasch mit der Komplexität der zu verarbeitenden Datenstrukturen. Insbesondere gilt dies, wenn zu ihrer Verwaltung die Organisation einer Speicherhierarchie erforderlich ist. Möglichst viele Details dieses Organisationsaufwandes für die Nutzer von Programmsystemen zu verdecken, ist eines der wesentlichen Ziele bei der Konzeption von Datenverwaltungssystemen. Bei virtuellen Speichern wird dies durch eine problemunabhängige Unterteilung des (virtuellen) Speicherraumes in Seiten meist gleicher Länge und einen in der VS-Steuerung fixierten Paging-Algorithmus erreicht, wobei eine ständige Speicherzugriffsüberwachung erfolgt. Für den Nutzer unsichtbar, sorgt die VS-Steuerung stets für die Anwesenheit der aktuell benötigten Teilstrukturen (Seiten) im Realspeicher. Entscheidende Nachteile ergeben sich jedoch bezüglich der Programmlaufzeiten bei einer zu freien Nutzung dieser Vorzüge eines virtuellen Speichers /7/. Eine willkürliche Unterteilung problemorientierter Datenstrukturen in problemunabhängige Teilstrukturen führt i. a. zu einer sehr hohen Transferrate. Deshalb wird als Qualitätskriterium für Programme in virtuellen Speichern deren Lokalitätsverhalten herangezogen, d. h. die Fähigkeit der Algorithmen, während des Aufenthalts einer Seite im Realspeicher deren Informationsgehalt maximal zu nutzen, bzw. über möglichst lange Zeiträume jeweils nur zu einigen wenigen Seiten zuzugreifen.
Daraus wird klar, daß für das Erreichen guter Lokalitätseigenschaften die Art und Weise der Abbildung problemorientierter Datenstrukturen auf die Seiten des VS und deren Berücksichtigung in den Algorithmen von grundlegender Bedeutung ist.
Deshalb kann es für ein gutes Laufzeitverhalten von Programmsystemen durchaus gerechtfertigt sein, einen "problemorientierten virtuellen Speicher" zu organisieren, indem die Steuerung der Speicherhierarchie aus der Ebene des Betriebssystems in die Ebene elementarer problemorientierter Verarbeitungsprogramme gehoben wird, auch wenn dadurch ein gewisser Entwicklungsmehr-

aufwand entsteht. Die dazu erforderliche Analyse der zu behandelnden Problem- und Datenstrukturen mit dem Ziel der Transferminimierung kann völlig neue, effektivere Lösungsvarianten hervorbringen, die sich besonders durch niedrigere Laufzeiten infolge eines guten Lokalitätsverhaltens auszeichnen. Schließlich kommt es noch darauf an, den Nutzer des Programmsystems nicht mit zu vielen Details der Datenverwaltung zu belasten. Dann ist es ihm auch gleichgültig, auf welcher Ebene der Programmhierarchie die Verwaltung der Speicherhierarchie stattfindet, da das Programmsystem insgesamt als "Black-Box" betrachtet wird /6/. Dazu trägt die ausschließliche Nutzung der expliziten Datenverwaltungsfunktionen auf der Ebene elementarer Verarbeitungsmoduln bei.

## 3. Parallelität in der Datenverwaltung

Bei der numerischen Verarbeitung sehr großer Datenmengen innerhalb einer Speicherhierarchie ist der erforderliche Zeitaufwand für Rechnung (CPU-Arbeit) und Transfer (I/O-Arbeit) nahezu von gleicher Größenordnung. Für eine effektive Datenverwaltung ist der Direktzugriff zu extern gespeicherten Daten eine Notwendigkeit. Um die dabei auftretenden Wartezeiten der ZVE zu verringern, wurde das von der sequentiellen Dateiverarbeitung bekannte Prinzip der Pufferung für den Direktzugriff verallgemeinert. Das ist gerechtfertigt, weil zwar die Zugriffsreihenfolge nicht mit der Reihenfolge der physischen Darstellung auf dem Speichermedium übereinstimmt, aber im Problemprogramm bereits einige Verarbeitungsschritte im voraus bekannt ist. Betrachtet man den Lösungsprozeß näherungsweise als eine Folge von Operationen $OP_k\ (E_k, A_k)$ über einer Menge $E_k$ von Eingangsdaten und einer Menge $A_k$ von Ausgangsdaten ($k = 1,2,\ldots,M$), so ist prinzipiell folgende Vorgehensweise möglich, wobei parallel auszuführende Aktionen durch // getrennt sind (vgl. /2/):

input $(E_1)$; $OP_1\ (E_1, A_1)$ // input $(E_2)$
for k := 2 to M-1 do
    $OP_k\ (E_k, A_k)$ // {output $(A_{k-1})$; input $(E_{k+1})$} ;
$OP_M(E_M, A_M)$ // output $(A_{M-1})$ ; output $(A_M)$;

Zur Realisierung dieser Parallelität wurde auf der Basis eines Zwei-Prozeß-Systems eine elementare Seitenverwaltung geschaffen, die mit nur wenigen einfachen Operationen dem Problemprogrammierer die Möglichkeit bietet, effektive problemorientierte Datenverwaltungen zu konstruieren /6/. Dabei wird der Speicherraum in "Seiten" einer problemabhängig wählbaren Länge unterteilt. Zur unmittelbaren Verarbeitung des Seiteninhalts muß sich dieser in einem "Seitenrahmen" im Hauptspeicher befinden. Die Anzahl dieser Seitenrahmen richtet sich nach dem aktuellen Bedarf und der verfügbaren Hauptspeicherkapazität.
Zur Verwaltung der Speicherhierarchie stehen im wesentlichen die folgenden Operationen zur Verfügung, wobei mit S ein Identifikator des logischen Speichermediums (z.B. Dateinummer, DD-Name), mit N eine Seitennummer und mit Z ein Zeiger auf einen Seitenrahmen bezeichnet wird:

- DASSGN (Z) : Zuweisen eines freien Seitenrahmens;
- DRELSE (Z) : Freigabe eines Seitenrahmens;
- DWRITE (S,N,Z) : Übertragung des Inhalts des durch Z angegebenen Seitenrahmens in die N-te Seite des Speichers S, ohne das Ende der Transferoperation abzuwarten;
- DPOINT (S,N) : Übertragung der N-ten Seite des Speichers S in einen freien Seitenrahmen, ohne das Ende der Transferoperation abzuwarten;
- DREAD (S,N,Z) : Zuweisen des Seitenrahmens, der die N-te Seite des Speichers S enthält, wobei ggf. das Ende einer Transferoperation abzuwarten ist.

Die beiden parallelen Prozesse (CPU und I/O) wurden als Haupt- und Unteraufgabe im Betriebssystem OS/ES simuliert. Eine vollkommene Parallelität ist so nicht erreichbar, da der I/O-Prozeß zum Starten der Transferoperationen jeweils ein geringes Quantum CPU-Zeit benötigt (in Konkurrenz zum CPU-Prozeß).
Die genannten Seitenverwaltungsoperationen können als FORTRAN-Unterprogramme aufgerufen werden, wurden jedoch aus Effektivitätsgründen selbst in der Assemblersprache realisiert. Für die durch FORTRAN nicht unterstützte Arbeit mit Zeigern wird eine einfache Simulation durch "Pseudo-Arrays" benutzt.

## 4. Anwendung

Dieses Seitenverwaltungssystem entstand im Rahmen der Behandlung großdimensionierter Matrixeigenwertprobleme, wo es seine erste Anwendung fand. Entsprechend dem Konzept der abstrakten Datentypen wurden für den Typ "schwachbesetzte symmetrische Matrix" elementare Matrixoperationen definiert, die für jede konkrete Speicherform der Matrix durch einen entsprechenden "Grundmodul" realisiert werden. Für den eigentlichen Lösungsalgorithmus (z.B. Eigenwertberechnung) bleibt damit die konkrete Speicherform ebenso wie die Organisation der Speicherhierarchie verborgen, er arbeitet nur über den abstrakten Objekten unter Verwendung der definierten Operationen /6/.
Die einzelnen Speicherformen für Matrizen unterscheiden sich in der Art und Weise der Bildung von Teilstrukturen und deren Abbildung auf die Seiten der Speicherhierarchie, wobei es im Interesse einer geringen Seitenwechselrate zweckmäßig ist, je eine logische Teilstruktur der Daten einer Seite vollständig zuzuordnen, auch wenn einzelne Seiten dabei nicht völlig genutzt werden /1/. Die Wahl der Speicherform richtet sich vor allem nach dem Besetzungsmuster der Matrizen (z.B. Bandstruktur, Zeilen- oder Spaltenprofil) und dient dem Ziel, Speicherplatz und unnötige Rechenoperationen einzusparen /5/.
Zur Minimierung der Transferzahl trägt auch wesentlich die Simultanisierung von Iterationsverfahren bei, d. h. eine Matrix wird stets auf mehrere Vektoren gleichzeitig und nicht nacheinander angewendet /4/.
Der Nachweis der durch Parallelität von Datenverarbeitung und -transfer erzielten Laufzeiteinsparung ist auf indirektem Wege möglich, indem die Meßwerte $t_{CPU}$, $t_{I/O}$, $t_L$ für CPU-Zeit, Transferzeit und Gesamtlaufzeit ausgewertet werden.
Wird mit $t_0$ die Zeitdauer echter Überlagerung von CPU- und I/O-Arbeit bezeichnet, so gilt offenbar (unter den Bedingungen des Multiprogrammbetriebs):

$$t_L \geq t_{CPU} + t_{I/O} - t_0 \,, \quad \text{d. h.}$$

$$t_0 \geq t_{CPU} + t_{I/O} - t_L = \tilde{t}_0 \,.$$

Die aus den Meßwerten berechenbare Größe $\tilde{t}_0$ stellt somit eine untere Schranke für den tatsächlichen Zeitgewinn dar.
Dieser Zeitgewinn kommt nicht nur dem betreffenden Programm selbst zugute, sondern gewährleistet eine insgesamt bessere Auslastung der wertvollen CPU-Zeit durch "Erschließung ungenutzter Reserven".
In /6/ wurden bereits Meßwerte für einige Grundmoduln angegeben, die später auch für zahlreiche weitere Anwendungsfälle bestätigt werden konnten. So ergibt sich für die Grundmoduln wie auch für ganze Eigenwertalgorithmen eine Einsparung $\tilde{t}_0$ in einer Größenordnung von etwa 30 bis 90 % der benötigten CPU-Zeit, wobei man davon ausgehen kann, daß die CPU-Zeit für einen gegebenen Modul und ein konkretes Beispiel charakteristisch ist, während die Transferzeiten durch zufällige Einflüsse des Multiprogrammbetriebes größeren Schwankungen unterliegen.

Literaturhinweise

/1/ Coffman, E. G., McKellar, A. C., Organizing Matrices and Matrix Operations for Paged Memory Systems, CACM 12, 3 (1969), 153-164

/2/ Hoare, C. A. R., Towards a Theory of Parallel Programming, in: Operating Systems Techniques, Academic Press, London 1972, 61-71

/3/ Lang, W., Ein Programmsystem zur Lösung großdimensionierter Eigenwertprobleme, ZfR-Information, INFO 81 Sektion V, AdW DDR, Berlin 1981, 3-8

/4/ Meyer, A., Über Algorithmen zur Lösung von großdimensionierten allgemeinen Eigenwertproblemen, ZfR-Information, INFO 81 Sektion V, AdW DDR, Berlin 1981, 9-15

/5/ Meyer, A., Döhler, B., Meinel, G., Pester, M., Dokumentation zum Programmpaket "Großdimensionierte Eigenwertprobleme", Teil 1, Technische Hochschule Karl-Marx-Stadt, Wissenschaftliche Informationen 32/1982

/6/ Pester, M., Eine Möglichkeit zur Effektivitätssteigerung bei der Verarbeitung großer numerischer Datenmengen durch Parallelisierung von Arithmetik und Transfer, WBZ MKR/IV TU Dresden 58/82, 80-89

/7/ Schiemangk, H., Virtuelle Speicher, Schriftenreihe Informationsverarbeitung, BSB B. G. Teubner Verlagsgesellschaft, Leipzig 1978

Verfasser:

Dipl.-Math. Matthias Pester
TH Karl-Marx-Stadt
Sektion Mathematik
9022 Karl-Marx-Stadt
Reichenhainer Str. 41

Fellin, Arne

# Zur Konzeption des Informationssystems der Deutschen Staatsbibliothek

## 1. Einleitung

Ebenso, wie in anderen Bereichen der Volkswirtschaft, besteht für das Bibliothekswesen der DDR die Notwendigkeit, eine Vielzahl von Arbeitsprozessen zu automatisieren. Diese geschieht vorrangig mit dem Ziel, die Versorgung mit wissenschaftlichen Informationen qualitativ zu verbessern und den Zeitraum bis zu ihrer Verfügbarkeit zu verkürzen. Während in der Information/Dokumentation bereits eine Reihe von Informationssystemen praxiswirksam geworden sind, nutzt das Bibliothekswesen nur vereinzelt Projekte, deren Effektivität durch ihre isolierte Stellung stark eingeschränkt ist. Die komplexe Automatisierung bibliothekarischer Prozesse verlangt jedoch noch die Lösung einiger Probleme, die im Folgenden kurz skizziert werden.

## 2. Inhaltliche Probleme

Die logische Architektur von Informationssystemen des Bibliothekswesens ist dadurch bestimmt, daß eine Vielzahl verschiedenartiger Daten von Informationsquellen (im Sinne der Datenverarbeitung) zu verwalten und zur Nutzung bereitzustellen sind. Im Gegensatz zu Inf./Dok.-Systemen, die sich meist auf eingeschränkte Klassen von Dokumenten konzentrieren, müssen Bibliothekssysteme unter anderem die Daten von Büchern, Zeitschriften, Zeitungen, Amtsdruckschriften, Kartenwerken, Musikalien, Microfilmen, Platten, Tonbändern, Bildern usw. erfassen, warten und verfügbar machen. Für die Zwecke der Sachkatalogisierung lassen sich die Klassifikations-, Deskriptor- und Thesaurussysteme, wie sie aus der Inf./Dok. bekannt sind, nützlich anwenden. Den bibliothekarisch wichtigsten Prozeß der alphabetischen Katalogisierung unterstützen sie aber nur ungenügend. Allein für Monographien und fortlaufende Sammelwerke sind dabei 800 Paragraphen der international verbindlichen Regeln für die alphabetische Katalogisierung einzuhalten. Zum großen Teil existieren auch für die anderen, oben genannten, Informationsquellen entsprechende international abgestimmte Vorschriften,

Auch die inhaltliche Erschließung von Bibliotheksgut bereitet noch große Probleme, da die Informationen praktisch alle Gebiete menschlichen Wissens überdecken. Das gilt besonders für große und mittlere Bibliotheken. Entsprechende Thesaury müßten weiterhin berücksichtigen, daß fremdsprachige Literatur in einer Vielzahl von Landessprachen in den Beständen von Bibliotheken anzutreffen ist. Dieses Problem ist zur Zeit weltweit nicht gelöst. Schließlich muß bemerkt werden, daß der Nutzerkreis von Bibliothekssystemen sehr heterogen ist und einer starken Fluktuation unterworfen. Eine Schulung ist folglich so gut wie ausgeschlossen.

## 3. Methodik

Die genannten Probleme stellen nur eine grobe Auswahl dar. Sie zeigen aber, daß sich die Entwicklung von einzelnen Projekten über lange Zeiträume erstrecken würde. Damit ist anzunehmen, daß nach ihrer Fertigstellung die zugrundegelegte rechentechnische und programmtechnische Basis nicht mehr verfügbar ist. Deshalb ist eine komplexe Projektierung anzustreben, die auf allen Ebenen ein hohes Maß an Portabilität gewährleistet. Weiterhin muß die Lösung auf spezielle Anforderungen der Textverarbeitung /2, 5/ zugeschnitten sein. Die Projektierung sieht deshalb als Kern des Informationssystems ein relationales Datenbanksystem vor, das Anforderungen der Textverarbeitung gerecht wird. Als theoretisches Modell wird eine Menge abstrakter Datentypen /5/ benutzt. Damit wird auf allen Ebenen eine klare Schnittstellendefinition gesichert. Zu ihrer Beschreibung wird die Systemprogrammiersprache CONCURRENT PASCAL verwendet, wodurch die wesentlichen Gesichtspunkte der Parallelverarbeitung abgedeckt werden können. Die Portabilität des Systems wird damit auf die von CONCURRENT PASCAL zurückgeführt. Einschränkungen ergeben sich aus dem Typkonzept der Sprache, das für die Verarbeitung von Textdaten schlecht geeignet ist. Deshalb wird zur Zeit an einer Erweiterung der an der Humboldt-Universität vorliegenden Version /6/ gearbeitet. Weitere methodische Gesichtspunkte gehen aus den folgenden Skizzen zur Architektur hervor.

## 4. Das Kommunikationssystem

Die Schnittstellen zwischen dem Nutzer und dem Datenverwaltungs-

system ist das Kommunikationssystem. Es realisiert die syntaktische Analyse, semantische Analyse und semantische Synthese für die Gesamtheit aller Nutzersprachen. Unter der semantischen Synthese ist dabei die Transformation in das Standardinterface $L_0$ (s. Abschn. 6) zu verstehen. Wegen des heterogenen Nutzerkreises von Bibliothekssystemen wird die Dialogführung so gestaltet, daß in Abhängigkeit von den Kenntnissen des Nutzers, dieser an die Lösung seines Problems herangeführt wird. Das wird während der syntaktischen Analyse dadurch erreicht, daß nach Beendigung einer Eingabe entschieden wird, ob ein

(1) vollständiger und syntaktisch korrekter
(2) unvollständig aber syntaktisch korrekter ,
(3) syntaktisch nicht korrekter

Satz vorliegt. Im Fall (1) wird der Satz sofort an die Verarbeitung weitergereicht. In den Fällen (2) und (3) wird automatisch eine "help"-Funktion /3/ ausgelöst, die vom Zustand der Analyse abhängt. Als "help"-Funktionen kommen Menü- und Formtechniken zur Anwendung, die den Nutzer über mögliche und notwendige Datenelemente informieren. Zu den Sprachen, die auf diese Weise behandelt werden, zählen außer den Recherchesprachen auch diejenigen, die bei der Datenerfassung Verwendung finden. Das hat seine Ursache darin, daß bibliographisch/bibliothekarische Daten keine standardisierte n-Tupeldarstellung gestatten. Wegen der Vielfalt zu verarbeitender Sprachen müssen zu ihrer Implementierung moderne Technologien zur Anwendung kommen.

## 5. Sprachimplementierung

Nutzersprachen werden automatisch durch das System LAISY (Language Implementation supporting System) /2/ implementiert. Als Eingabesprache für LAISY wird eine Sprache zur Beschreibung von attributierten Grammatiken verwendet, die es gestattet kontextfreie Sprachen syntaktisch und semantisch zu definieren. Aus jeder Sprachbeschreibung wird eine Tabelle zur Steuerung der syntaktischen Analyse und ein Interpreter für die semantische Analyse und Synthese erzeugt. Der semantische Analyseteil des Interpreters kann so erzeugt werden, daß auch Informationen des Data-Dictionary in die semantische Analyse einbezogen werden können. Zweckmäßig ist z.B. für die frühzeitige Kontrolle von Zugriffs-

rechten, der Existenz angesprochener Relationen aber auch für Integritätskontrollen. Die Einbettung von LAISY in das Informationssystem gestatten eine flexible Anpassung an neue Anforderungen und gewährleistet damit die Erweiterungsfähigkeit auf der Kommunikationsebene.

## 6. Das Standardinterface $L_0$

Im Ergebnis der semantischen Synthese von o.g. Interpretern entsteht eine Baumstruktur. Die Gesamtheit so erzeugter Bäume wird als das Standardinterface $L_0$ bezeichnet. Er repräsentiert ein subset von SEQUEL /1/. Als Voraussetzung wird deshalb angenommen, daß auf der externen Ebene nur solche Sprachen zur Anwendung kommen, die eine eindeutige Interpretation in SEQUEL bzw. einer Untermenge davon gestatten. Die SEQUEL-Repräsentation von $L_0$ wird auf der externen Ebene dem Datenbankadministrator (als einzigem Nutzer dieser Sprache) zur Verfügung gestellt. Dieser nutzt sie zur Wartung der Datenbasis, des Data-Dictionary und zur Realisierung von umfangreichen Recherchen. $L_0$ ist demgemäß Schnittstelle zwischen dem Kommunikationssystem und dem Datenverwaltungssystem. Die Elemente von $L_0$ werden im Datenverwaltungssystem so transformiert, daß sie durch den $L_0$-Interpreter abgearbeitet werden können. Die Transformation schließt das Bestimmen von Zugriffspfaden, Zugriffsoptimierung und Integritätskontrollen ein.

## 7. Data-Dictionary

Das Datenverwaltungssystem unterscheidet zwischen virtuellen und physischen Relationen /4/. Virtuelle Relationen sind auf der $L_0$-Ebene angesiedelt. Sie umfassen virtuelle Basisrelationen, Links und Selektionen. Jeder virtuellen Relation ist ihre Realisierung in Gestalt einer Menge physischer Relationen zugeordnet. Das Data-Dictionary enthält deshalb neben den üblichen Informationen wie Zugriffsrechte, Attributnamen, Attributtypen, Zustand der Relation ..., alle Informationen über die physische Ausprägung. Damit ist die Abbildung von virtuellen Entities auf physische umkehrbar eindeutig definiert. Als physische Relationen werden im Datenverwaltungssystem Wurzelrelationen, Deskriptorrelationen, DA-Relationen und SQ-Relationen /Details in 4/ unterstützt. Diese Aufteilung ist an die Notwendigkeit angepaßt, Texte als Bäume zu speichern und eine Differenzierung nach Zugriffshäufigkeit

vorzunehmen.

8. Schlußbemerkungen

Von den Komponenten des geplanten Systems konnte hier nur eine Teilmenge grob umrissen werden. Tiefergehende Aussagen sind in der systembezogenen Literatur zu finden. Eine Pilotimplementierung ist für Ende 1983 geplant. Etwa 1985 kann das System für Nachnutzer bereitgestellt werden.

Literatur

/1/ Astrahan, M.; Chamberlin, D.: Implementation of a structured English Query Language. In: Communications of the ACM, 18(1975)10

/2/ Fellien, A.: Grundsätze der Implementierung von Textverarbeitungssystemen. In: Rechentechnik-Datenverarbeitung, Berlin, 18(1981)9

/3/ Kreutzer, B.: Ausgewählte Probleme des Kommunikationsteils des DBBS der DSB. Vortrag zum 4. Betriebssystemseminar der HFR/IV. Zechlin, 1982 (unveröffentlicht)

/4/ Müller, W.: Aufbau eines Data-Dictionary im Rahmen des Datenbanksystems der DSB. Vortrag zum Problemseminar Datenbanken. Gaußig, 1982

/5/ Walch, G.; Koch, U.: Abstrakte Datentypen als Hilfsmittel für die Implementierung eines DBMS. Vortrag zum 10. Kolloquium über Rechentechnik und Datenverarbeitung. Magdeburg, 1982

/6/ Werner, H.: Das concurrente Pascalprogramm SOLO ... . In: Wissenschaftliche Zeitschrift der Humboldt-Universität. Berlin, (1980)4

Verfasser:
Dipl.-Math. Arne Fellien
Deutsche Staatsbibliothek Berlin
Organisations- und Rechenzentrum
1086 Berlin, Unter den Linden 8

Wagner, Klaus

# Zum Problem der Anwendung der Datenbanktechnologie

## 1. Problemstellung

Die Anwendung der Datenbanktechnologie erfolgt weltweit in geringerem Tempo als ursprünglich erwartet. Im Mittelpunkt dieser Technologie steht der Einsatz solcher universeller Systemunterlagen bzw. Datenbankbetriebssysteme (DBBS) wie ADABAS, DATACOM/DB, DBS/R, IDMS, IDS, IMS usw. (weltweit ca. 200 Systeme). Hauptzweck des Einsatzes von DBBS ist die gemeinsame Benutzung von Datenelementen, Segmenten, Sätzen oder Dateien durch viele (EDV-)Benutzer mit unterschiedlichen Anwendungsprogrammen auf der Grundlage einer einheitlichen, gemeinsamen Benutzerschnittstelle: dem Datenbankschema. Jeglicher Zugriff erfolgt in den Termen des Datenbankschemas.

Die Mehrheit der Aussagen zum Pro und-Kontra der Anwendung der Datenbanktechnologie - in der EDV-Praxis oft mit "Datenbank-Würdigkeit" umschrieben - sind unbefriedigend. Die langjährige Abstinenz wissenschaftlicher Betätigung auf diesem Gebiet bedeutete für zahlreiche Betriebe und für die in der EDV engagierten Mitarbeiter mitunter kostspielige Lernprozesse.

Im folgenden werden die verschiedenen Aspekte der Anwendung der Datenbanktechnologie aus der Blickrichtung der Betriebsinformatik (als der Wissenschaft vom Aufbau, der Arbeitsweise und Gestaltung automationsunterstützter betrieblicher Informationssysteme) dargelegt. Ziel des Beitrages ist eine kritische Bestandsaufnahme der gegenwärtigen Aussagen zum Problem der "Datenbank-Würdigkeit" und die Ableitung von Schlußfolgerungen, die die weitere Arbeit zur Schaffung von Grundlagen der Betriebsinformatik betreffen.

## 2. Stand der gegenwärtigen Aussagen

Im folgenden sollen die von einigen Autoren genannten Vorteile der Anwendung von DBBS stichpunktartig angeführt werden:

H. WEDEKIND (1974):

- Zentralisation der Dateien des automationsunterstützten betrieblichen Informationssystems
- Realisierung eines hohen Grades an Datenunabhängigkeit

D. S. KOREIMANN (1974):

- Rationalisierung der Datenpflege
- Integration bisher unabhängiger EDV-Anwendungen

C. J. DATE (1976):

- zentrale Steuerung und Überwachung der verfügbaren operationalen Daten (des Betriebes)
- Vermeiden von Inkonsistenzen
- Vielfachverwendbarkeit (sharing)
- Standardisierung
- Zentrale Sicherheitsvorkehrungen
- Gewährleistung der Datenintegrität
- Vermeiden von Zugriffskonflikten
- Datenunabhängigkeit

W. ABT (1980):

- Senken der Redundanz
- Speicherplatzeinsparung
- Vereinfachung der Pflege der Daten
- Vermeiden von Sortierungen
- Nutzer kann im Datenbestand ohne tiefere Kenntnisse recherchieren
- Zugriffsberechtigung wird automatisch überprüft
- Reorganisation wird vom DBBS übernommen

D. S. APPLETON (1980): prägte das AKRONYM "IRRIASSPA":

I - data independence
R - relatability
R - non-redundancy
I - integrity
A - accessibility
S - shareability
S - security
P - performance
A - administration

Hervorzuheben sind die Untrersuchungsergebnisse von G.K. und J.J. Wiorkowski /1/, die sich auf eine Umfrage auf der Basis von 27 Unternehmen/Anwendern der DBBS ADABAS (4x), DATACOM/DB (5x), IMS (8x), SYSTEM 2000 (5x) und TOTAL (5x) stützen. Im Ergebnis der Umfrage entstand folgende Liste der Vorteile

(Rangfolge ist zu beachten!):

1. Datenunabhängigkeit
2. Datenintegrität
3. Online-Unterstützung
4. Zentralisierte Kontrolle der Dateien
5. Geringer Aufwand und hohe Flexibilität bei der Restrukturierung und Pflege der Daten
6. Verringerung der Datenredundanz
7. Integrierte statt unabhängige EDV-Anwendungen
8. Schnelles Reagieren auf nichtvorhergesehene Informationsbedürfnisse
9. Anwendungsprogrammierer benötigen keine Kenntnisse über die physische Struktur der Daten
10. Datensicherheit und Datenschutz (security and privacy)

Datenunabhängigkeit, Datenintegrität und die Implementierung von Online-Anwendungen (Echtzeitprojekte) nehmen bei den Praktikern gegenwärtig eine Schlüsselstellung ein. Die redundanzarme Speicherung - neben der Speicherung der logischen Beziehungen zwischen den Daten und dem direkten Zugriff häufig als die Merkmale einer Datenbank genannt /2/ - rückte auf den 6. Platz.

Zur Unterstützung der Entscheidung darüber, ob bei der Automatisierung einer Datenverarbeitungsaufgabe (bzw. Aufgabenklasse) ein DBBS eingesetzt werden sollte oder nicht, wurde eine Reihe von Kriterienkatalogen bzw. Kriterienübersichten entwickelt:

VIGIER (1971), HUGENBERG und KAISER (1972), PLESCH und GRIESE (1972), LORENZ (1974) ADAM (1976), OSSWALD (1977), ERFA PPS Zürich (1977), KOSCHE (1980).

Kritisch ist anzumerken:

1. Die Mehrheit der Kataloge/Übersichten wurde vorrangig unter dem Aspekt des Einsatzes eines speziellen DBBS oder der Auswahl unter verschiedenen DBBS entwickelt.
2. Die Kataloge/Übersichten gründen sich auf konkrete Datenverwaltungsanforderungen bzw. Datenverarbeitungserfordernisse und orientieren allein auf die Effizienz der EDV-Prozesse (Faktor Zeit als einziger Maßstab!). Sie unterstützen

demzufolge weder die Einführung neuer Organisationsformen der EDV noch die Erschließung neuer EDV-Anwendungsgebiete. Die Zugrundelegung des Faktors Zeit ist prinzipiell richtig, muß aber auf die Leitungsprozesse und Evolution des gesamten automationsunterstützten betrieblichen Informationssystems ausgedehnt werden (Notwendigkeit des Unterscheidens zweier Nutzenskategorien /2/).

3. Die Kataloge/Übersichten enthalten trotz eingeengter Herangehensweise an das Problem der "Datenbank-Würdigkeit" keine quantitativen Aussagen (Meßlatten).

## 3. Wie sollte die Lösung des Problems erfolgen?

Aus der Blickrichtung der Schaffung von Grundlagen der Betriebsinformatik sollte das Problem der "Datenbank-Würdigkeit" in zweifacher Weise angegangen werden:

### 1. Lösungsrichtung

Ermittlung der Kriterien der Anwendung der Datenbanktechnologie für die betrieblichen Teilsysteme (horizontal-funktionale und vertikal-hierarchische Gliederung der betrieblichen Aufbauorganisation).

Lösungsweg:
- Bildung von Klassen/Typen automatisierter Informations(teil)systeme
- Ermittlung des Zusammenhangs von Datenbanktechnologie und der Implementierung der Informations(teil)systeme
- Zuordnung der Informations(teil)systeme zu den betrieblichen Teilsystemen gem. Aufbauorganisation

Folgende Entwicklungstendenzen sind zu berücksichtigen:
- zunehmende rechnergestützte Entscheidungsvorbereitung im Leitungs- und Planungsprozeß
- verstärkte Anwendung der EDV in Forschung, Entwicklung sowie zur Projektierung, Konstruktion und technologischen Vorbereitung
- zunehmende Integration der EDV-Prozesse
- Dialogisierung vorhandener Stapelverarbeitungs-Anwendungen

Der Zusammenhang von Aufbauorganisation und Informations(teil)-systemen wird in Abb. 1 skizziert. (Vgl. auch die Dissertationen von H. Döringer (Mannheim 1975) und K. Garbe (Dresden 1976).)

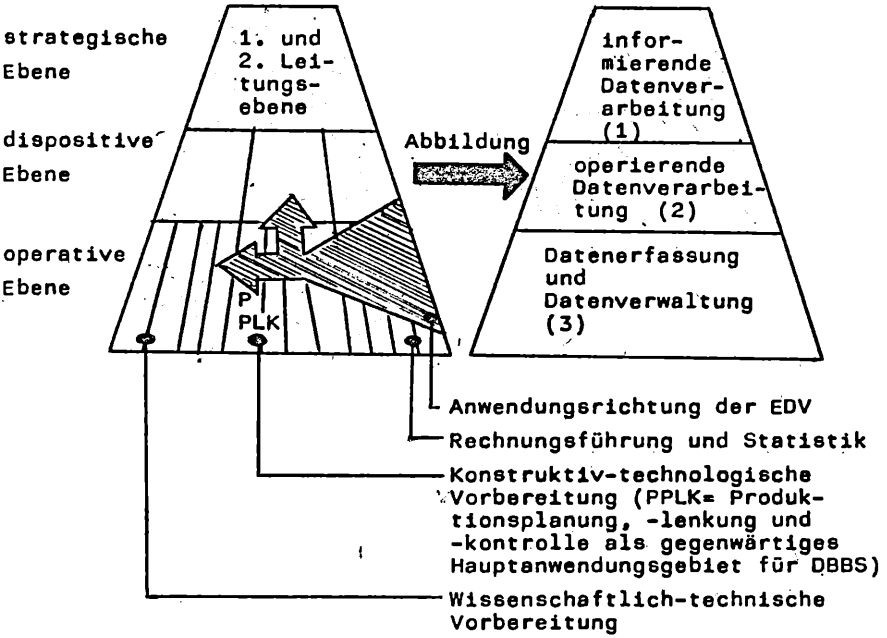



Abb. 1: Betriebliche Aufbauorganisation und Informations-(Teil)systeme

In Abb. 1 bedeuten: (1) Planungs- und Entscheidungssysteme
(2) Berichts- und Kontrollsysteme
(3) Transaktionsdatensysteme

## 2. Lösungsrichtung

Quantifizierung der für die Anwendung der Datenbanktechnologie wichtigsten Kriterien (Meßlatten-Problem).

Lösungsweg:
- Definition von Maßstäben für die wichtigsten Datenbankeigenschaften
- Überprüfung der Metrik an ausgewählten DBBS
- Entwurf einer Metrik für Informationssysteme auf Grundlage eines theoretischen Modells
- Gewinnung von Relationen zwischen beiden Metriken

Folgende Entwicklungstendenzen sind zu berücksichtigen:

- zunehmende Bedeutung der Datenunabhängigkeit als Grundlage für die
  . Evolution des Informationssystems (Stabilität trotz Wachstum!)
  . Erschließung neuer Nutzerklassen aus den Fachabteilungen
  . Realisierung der dynamischen Informationsverarbeitung /3/
- zunehmende Bedeutung von Flexibilität bei gleichzeitigem Zwang zur Standardisierung

## 4. Einige Erkenntnisse (Thesen)

1. These: Massendatenverarbeitung

Wenn für die Automatisierung einer Datenverarbeitungsaufgabe lediglich (zentraler) Stapelbetrieb möglich bzw. nur diese Organisationsform der EDV sinnvoll ist, dann ist es schwierig, die Vorteile der Anwendung der Datenbanktechnologie aufzuzeigen.

Ausnahme: Kurzfristige Produktionsplanung, -lenkung und -kontrolle (Stücklistenspeicherung)

Kriterien:
- Effizienz der Ausführung (insbesondere bei selektiven Auswertungen mit bis zu 5 % Aktivierungsrate der Datensätze in verschiedenen Dateien (Netzstrukturen) /4/) [1]
- Integration von EDV-Anwendungen [2]
- Datenunabhängigkeit (als Schutz existierender Anwendungsprogramme vor Änderungen infolge von Veränderungen in den Dateien)
- Flexibilität
- Standardisierung (der Datenverarbeitungsprojektierung und Abarbeitung im Rechenzentrum)

---

1) Für entsprechende Oberlegungen ist u.a. die Zugriffsleistung in Sätzen pro Minute relevant: Datenbank: 100, Direktzugriff auf Platte: 1000, sequentieller Zugriff: 10000.

2) Die Integration besitzt mehrere Aspekte, vgl. /5/.

2. These: Interaktive Verarbeitung

Der entscheidende Grund für die Anwendung der Datenbanktechnologie ist gegenwärtig die Implementierung von EDV-Anwendungen in Echtzeitbetrieb, genauer: Teilhabersysteme. Dazu gehören:

- Datensammelsysteme (auf der Ebene der Datenerfassung und -verwaltung)
- Buchungs- und/oder Direktabfragesysteme, dispositive Sachbearbeiterfunktionen in programmkontrollierter Dialogverarbeitung (auf der Ebene der operierenden Datenverarbtg.)
- Entscheidungs-Unterstützungs-Systeme und Auskunftssysteme für die nutzerkontrollierte Dialogverarbeitung (auf der Ebene der informierenden Datenverarbeitung)

Kriterien:
- dezentraler (direkter) Zugriff auf zentralisierte Dateien
- Effizienz der Ausführung
- Datensicherheit und Datenschutz
- Datenunabhängigkeit
- Standardisierung

3. These: Neue Aufgabenklassen

Künftige Hauptrichtung der Anwendung der Datenbanktechnologie ist die rechnergestützte Entscheidungsvorbereitung in Gestalt von

1. operativen Informationssystemen (hervorgehend aus der Verschmelzung von vergangenheitsbezogenen und zukunftsbezogenen Systemen) zur Lösung kurzfristiger Informations- und Entscheidungsprobleme auf der dispositiven und operativen Ebene und

2. Entscheidungs-Unterstützungs-Systemen zur Lösung langfristiger, nicht-programmierbarer (schlecht-strukturierter) Entscheidungen auf der strategischen Ebene.

Kriterien:
- Zentralisation der Dateien/Integration der EDV-Anwendungen
- zentraler (direkter) Zugriff auf dezentrale Dateien (Datenbanksystem, verteilte Datenbank, Datenbankverbund /5/)
- Datenunabhängigkeit (als funktions- und aufgabenunabhängige Speicherung /3/)
- Flexibilität
- nutzerkontrollierter Dialog

## 5. Weiterführung der Arbeiten

Die Datenbanktechnologie ist das geeignete Instrumentarium für die flexible Automatisierung, d.i. Automatisierung ohne Starrheit zu erzeugen. Die Ermittlung der Kriterien für die Anwendung der Datenbanktechnologie muß auf der Evolution des automationsunterstützten betrieblichen Informationssystems, der Einführung neuer Organisationsformen der EDV und der Erschließung neuer EDV-Anwendungsgebiete, basieren /5, 6/. Es ist notwendig, diesbezüglich theoretische Grundlagen zu schaffen. Hiervon ausgehend und unter Berücksichtigung der Verantwortung des Hochschulwesens für die Grundlagenforschung wird daran gearbeitet, die aufgeworfenen Probleme - insbesondere die in Abschn. 3 formulierten - schrittweise einer (theoretischen) Lösung zuzuführen.

## Literatur

/1/ G.K. Wiorkowski, J.J. Wiorkowski: Does a Data Management pay off?, in: Datamation 24(1978)4

/2/ Diskussion zu "Datenbankerfahrungen", in: Rechentechnik/Datenverarbeitung 19(1982)7

/3/ K. Fuchs-Kittowski u.a.: Informatik und Automatisierung, Band 1, Berlin 1976

/4/ G. Wiederhold: Database Design, New York 1977

/5/ K. Wagner: Gestaltungs- und Entwicklungsziele künftiger Systemunterlagen für Datenbanken - Grundfragen ihrer Realisierbarkeit, Diss., HUB, Berlin 1981

/6/ V. Lange: Methodische Probleme und Verfahren der Projektierung von Datenbasissystemen, Diss., HUB, Berlin 1982

Verfasser: Dr. Klaus Wagner
Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum
1086 Berlin
PSF 1297

Bazewicz, Mieczysław, and Dębowy, Julian

# Software organization and utilization method in a medical information system

## 1. Introduction

In the process of designing a large computer application system the influence of different factors on system efficiency is investigated. Beside the economic criterion, a number of efficiency criteria, depending above all upon the destination of the system being designed are used in those considerations [1]. However, in all investigations an essential role is played by the degree of adaptation of the given system to specific requirements of its users. Such an adaptation depends mainly on the method of task formulation by the users, accepted in the system, as well on the properties of system software.

When evaluating system software, the following software features are mostly /e.g. [3]/ considered:

- correctness, understood as to be the degree of fulfilling user's goals,
- reliability, or the degree, to which system programs fulfill their functions with the required precision,
- usability, or expenditure for teaching system software use and service,
- efficiency, understood as to be the amount of computer resources used for handling and executing user's task.

In order to evaluate the possibility and easiness of modifying system functions, the following properties of system software are taken into consideration:

- maintainability, or the effort required to locate and fix an error in each of the system programs,
- testability, enderstood as to be the effort to test a program to insure it performs its intended functions,
- flexibility, understood as to be the effort required to modify an operational program.

In the evaluation of the system software design an important role is performed by the amount of expenditures necessary to its elaboration.

In the case of evaluating system adaptation to user's needs a great influence is exerted also by the construction of the language, in which the system user has to prepare his task.

The above mentioned remarks have been utilized during the development of the medical investigation system BAMED.

## 2. Characteristics of the medical investigation system BAMED

System BAMED is destined to store and process descriptions of health status of men, the descriptions being collected during the consecutive visits of patients to physicians.

The health status of a patient is described in the system by a set of values of features obtained as a result of medical examinations and laboratory tests. The results of these examinations are gathered in specially elaborated medical documents which permit to compile a complete medical profile of a given patient. In the description of the patient's health status different features emerge, which take logical, numerical, and alphanumerical values. The features are grouped into blocks. In the system four basic blocks groups can be distinguished, covering:

- anamnesis,
- physical medical examinations,
- laboratory tests,
- medical diagnosis.

In the data base system the values of features of each block are stored in the form of one record. Thus the description of the patient's health status stored in a compact form, can be represented as a set of records.

The basic feature of the system is the possibility of following changes, occuring in the time, of the health status of a population and of each of the patients. The use of the system is particularly suitable for performing prophylactic investigations of groups of persons working in conditions being noxious for the health, as the system permits to study the influence of the environment on the health status of the investigated populations. The system permits, to the physician, to obtain important epidemiological information, so it can contribute to eliminate a number of factors influencing noxiously the human health status.

The equipment of the BAMED system with a packet of programs of mathematical statistics, assures the system to be a convenient

tool for verifying medical hypotheses, e.g. concerning the causes of changes occuring in the health status of the investigated populations.

## 3. General description of the language for task formulation

The BAMED system users write their tasks in the form of a program in a problem oriented language, which has been elaborated during the system design process.

The task, written in this language, is a sequence of directives, instructions and parameters.

A directive is referred to as a sentence and serves as an information control for the translator of the language. When utilizing the problem oriented language, the following directives are employed:

- PROGRAM,
- STATISTICS,
- RETRIEVAL,
- UPDATING,
- PERFORMING,
- END,
- PRINTING.

The language parameters serve to determine the data object of each of the segment /in the program, three different segment types: statistical, retrieval and updating can appear/, and each instruction.

The parameters are used to determine:

D - the period/s/ of examination of the patients,

Z - the identifier of the patient or patient's group,

A - the list of features. A can appear in one of the following forms

$$(1) \qquad A = \begin{cases} \text{feature list;} \\ \text{feature list; logical condition;} \\ \text{: logical condition;} \end{cases}$$

where:
- the feature list is a sequence of block numbers and feature numbers presented in blocks,
- the logical condition can be simple or complex. If the first condition is fulfilled by the patient's data, it causes that data are introduced from the list into the patient's group the features of which are processed in the segment of program. The complex logical condition divides a patients group into subgroups.

- the appearance of a logical condition without feature list is utilized for obtaining the output of patients' number from the group defined by parameter Z, the data of which fulfill that condition.

Parametr C is designed for determining the features processed by a given segment. This parameter takes one of the forms mentioned in (1), but a logical condition, in parameter C, can be a simple logical condition, only. The set of features given in the features list and in the logical condition of each instruction must be a subset of the features present in the feature list of the given subset.

In Fig. 1 a graphic interpretation of a simple example of a complex logical condition is presented, that condition being given by formula (2)

(2) $CLC=(LC_1^{(1)},LC_2^{(1)}(LC_1^{(2)},LC_2^{(2)},LC_3^{(2)}))$

where: CLC - complex logical condition,
LC - simple logical condition.

Condition (2) is equivalent to the 6 following simple logical conditions:

- $LC_1^{(1)}$ AND $LC_1^{(2)}$,
- $LC_1^{(1)}$ AND $LC_2^{(2)}$,
- $LC_1^{(1)}$ AND $LC_3^{(2)}$,
- $LC_2^{(1)}$ AND $LC_1^{(2)}$,
- $LC_2^{(1)}$ AND $LC_2^{(2)}$,
- $LC_2^{(1)}$ AND $LC_3^{(2)}$.

In a complex logical expression two logical operators can appear: AND, OR, and in a simple complex expression only relation operators can appear:

EQ - Equal,
NE - Not Equal,
GT - Greater Then,
GE - Greater or Equal,
LT - Less Than,
LE - Less or Equal.

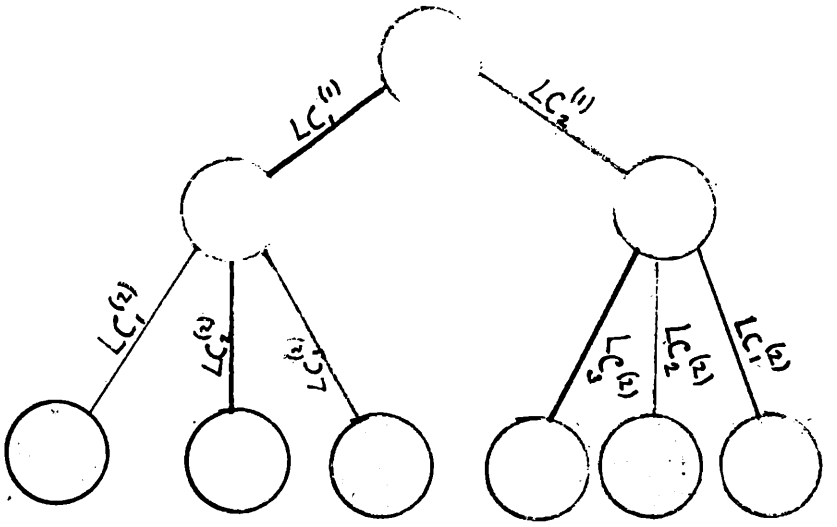



Fig. 1. Interpretation of a complex logical condition of form (2).

Example 1. A simple program, including one retrieval segment with one instruction of result output.

```
#P      /program/                  } directives
#W      /retrieval/
D=10.07.81/20.10.81;                                          }
Z=ZD(RKØ1,PRØ2),                                              } parameters
C=101(4,8-9),111(1-2),312(23),312(48):1Ø1Ø9 EQ Ø;             }
WYØ3                                          } instruction
A=1Ø1(4,8):(111Ø1 EQ 1,111Ø2 EQ 1(31223 GT 1ØØ,31248 GT 22Ø)); } parameter
#R      /perform/                  } directives
#K      /end/
```

The Example 1 program presents four lists of male workers /mans/ of departaments RKØ1, PRØ2 of plant ZD. Assuming that the meaning of block and feature numbers are as follows:

10104 - first name and surname,

10108 - home adress,

10109 - sex,

10109 EQ 0 - means that the patient is a man,

11101 EQ 1 - means that the patient does not smoke cigarettes,

11102 EQ 1 - means that the patient smokes more than 20 cigarettes per day,

31223 - sugar content in the blood,
31248 - cholesterol content in the blood.

The first and second lists, put out by the system include first names, surnames, and adresses of non-smoking patients, in whose blood a sugar level greater than 100 mg% and a cholesterol level greater than 220 mg% were detected. The third and fourth list include identical data of workers who smoke more than 20 cigarettes a day.

## 4. Software of the BAMED system

In the design of BAMED system software an important problem appeared to be the selection of its structure. That selection was influenced mainly by the following system features:

- a great number of different block types /in the present version the possibility of appearing 74 different blocks, among which 33 different record types is admitted/,
- the assumption, accepted a priori, of necessity of modifications of the system software, the modifications being connected first of all with the possibility of increasing the number of blocks in the description of man's health status, collected in the system, as well as with adding to the system new procedures, e.g. new algorithms of mathematical statistics,
- the assumption of elaborating certain programs of the system by students of our faculty.

In order to facilitate system software elaborating, and to facilitate system running, the following system structure, presented in Fig. 2, has been accepted:

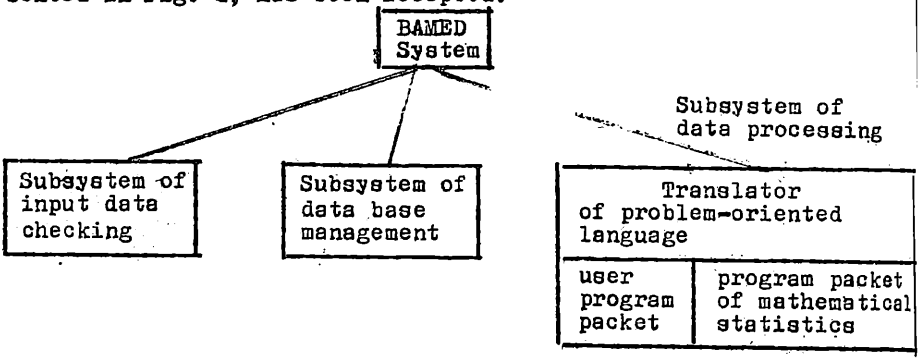



Fig. 2. General structure of the BAMED System.

The division of the system into subsystems, and the use of a modular structure of the software, permitted to obtain a detailed design of all the program modules of the system, as well as of their interrelations. So the process of elaboration of all program modules was facilitated, and their number could be reduced.

In order to improve the system efficiency, the software of selected modules was elaborated in assembler language, among which most of the modules of the subsystem of data input and checking and the data base management subsystem, as well as the language translator. The elaborated software of the BAMED System is very efficient, and it has all the features characterized in the first part of the present paper /introduction/. However, the fact that some of the program modules are elaborated in assembler language makes it impossible to utilize directly the system on another type of computer.

## 5. Conclusions

The designed and elaborated method of task formulation in the BAMED System fulfills all the necessary conditions of a language of an open system. This is of great practical importance, as the implementation of the system permits the creating of large files of information concerning the patients health over a span of years. The data can also be employed both for building mathematical models for solving selected medical problems and for determining the symptoms of particular diseases. This is necessary for elaborating methods of computerized medical diagnosis asisting the physician. Continuous supplementation of the language with new procedures will be necessary to achieve this goal.

The software design in a modular form has facilitated its development by a numerous designers team, and, in a great measure, by students, as it was possible to concurrently code and test many modules. The introduction of modifications to the software is also easy.

References:

[1] Bazewicz M., Systemy informatyczne w szkole wyższej, PWN, Warszawa 1980.

[2] Dębowy J., User task of the Bamed medical investigations system. W: materiałach Konferencji "Komputerowe Przetwarzanie Informacji Biomedycznej" /in preparation/.

[3] Walters G.F., Mc Call J.A., Software quality for life-cycle cost reduction, IEEE Trans. Reliability, 28, No 3, 1979.

Authors:
Ass.prof.dr. Mieczysław Bazewicz
Dr. Julian Dębowy
Technical University of Wrocław, Poland
Institute of Control and Systems Engineering
50-370 Wrocław
Wybrzeże Wyspiańskiego 27

Hagen, Manfred
Klaembt, Egbert
Schirmbacher, Peter

# Der Einsatz von automatisierten Informationsverarbeitungssystemen (AIVS) für den Leitungsprozeß

## 1. Einleitung

Unbestrittenermaßen stellt die ständige Vervollkommnung der Prozesse der Leitung und Planung ein wesentliches Erfordernis in unserer heutigen Zeit dar. Dies wurde auf dem X. Parteitag der SED hervorgehoben und, um nur wenige Beispiele anzuführen, läßt sich begründen mit der ständigen Zunahme des Umfanges und der Vielfalt der Produktion, der Vertiefung der gesellschaftlichen Arbeitsteilung und der Zunahme des Produktionspotentials sowohl in hochqualifizierten Arbeitskräften als auch modernen Arbeitsmitteln.
Mit der Erhöhung des Niveaus der Leitungstätigkeit kommen jedoch auch erhöhte Aufgaben zur Unterstützung dieser Arbeit auf die elektronische Datenverarbeitung zu. Dabei wird es in der Zukunft nicht in erster Linie mehr darum gehen, die langfristig beständigen Aufgaben des Leitungs- und Planungsprozesses zu bearbeiten, da diese im wesentlichen durch die bereits existierenden EDV-Projekte unterstützt werden, sondern vielmehr darum, zeitlich begrenzte, schwer oder nicht vorhersehbare Aufgaben, die sich erst innerhalb des Problembearbeitungsprozesses ergeben, in den Mittelpunkt der weitergehenden EDV-Anwendungen zu stellen. Diese nichtschematischen Aufgaben müssen in großer Zahl und kurzer Zeit bewältigt werden. Das verlangt eine wesentliche Erhöhung der Auskunftsfähigkeit und Reaktionsgeschwindigkeit der EDV. Dabei geht es nicht darum, die Effizienz der EDV-Anwendung in erster Linie zu erhöhen, sondern ihre Effektivität, d.h. die erreichte Verbesserung der Leitungstätigkeit muß Maßstab sein und nicht die günstigste rechentechnische Lösung.

Dieses Kriterium in den Vordergrund stellend, ergeben sich für die Bearbeitung nichtschematischer Aufgaben zwei Fragestellungen:

1. Wie wird die Kommunikation zwischen dem AIVS und dem Nutzer gestaltet?
2. Wie muß ein AIVS beschaffen sein, um die Anforderungen, die sich durch die Bearbeitung nichtschematischer Aufgaben ergeben, erfüllen zu können?

## 2. Die Gestaltung der Kommunikation zwischen dem Nutzer und dem automatisierten Informationsverarbeitungssystem (AIVS)

Eine notwendige Voraussetzung für die Effektivität der EDV für die Prozesse der Leitung und Planung bildet eine möglichst vollständige Integration der automatisierten Informationsverarbeitung in das gesamte Informationsverarbeitungssystem der Leitung und Planung. Aus diesem Grund kommt der Gestaltung der Schnittstelle zwischen dem Nutzer, wobei damit auch im weiteren der Informationsendverbraucher gemeint ist, und dem AIVS eine besondere Bedeutung zu. Wir halten es deshalb für erforderlich, sowohl die möglichen Mensch-Maschine-Kommunikationsformen als auch die vorherrschenden Aufgaben der Leitung und Planung näher zu charakterisieren.

Im allgemeinen unterscheidet man die direkte und indirekte Mensch-Maschine-Kommunikation.
Kennzeichnend für die direkte Mensch-Maschine-Kommunikation ist, daß der Nutzer seine Anforderungen an das jeweilige AIVS selbständig in einer entsprechenden formalen Sprache formuliert.
Im Gegensatz dazu kommuniziert der Nutzer bei der indirekten Mensch-Maschine-Kommunikation mit einer Gruppe von EDV-Spezialisten, von uns als Informationszentrale (IZ) bezeichnet, der er seine Anforderungen in der natürlichen Sprache übermittelt.
Die Informationszentrale setzt diese $A_n$forderungen in eine formale, maschinenverständliche Sprache um, leitet sie dem Rechner zu und übergibt dann die Ergebnisse in der gewünschten Form dem Nutzer.

Die direkte Kommunikation verlangt somit zusätzliche Kenntnisse vom Nutzer für den Umgang mit dem AIVS. Neben dem Hauptproblem des Nutzers, welche Daten sind wo und wie gespeichert und welche Mittel ihrer Auswertung stehen ihm zur Verfügung, können sich weitere Fragestellungen für ihn entwickeln. Z.B.: Wie ist der Zugang zur EDVA im jeweiligen Rechenzentrum organisiert? Welche Komponenten des Anlagenbetriebssystems werden zur Abarbeitung des Auftrages benötigt? Wie allgemein bekannt, sind heutzutage niveauvolle Softwaresysteme in der Lage, den Nutzer dabei wesentlich zu unterstützen. Dies trifft aber in den meisten Fällen nur für die Bearbeitung von schematischen Aufgaben zu. (Eine schematische Aufgabe ist die Konzipierung eines gleichbleibenden Schemas, nach dem die Abarbeitung der formulierten Aufgabe erfolgt. Es sind Aufgaben, die sich häufig und regelmäßig wiederholen und immer in der gleichen Weise ablaufen.) Auf Grund des begrenzten Datenvolumens und der mit großer Häufigkeit wiederkehrenden relativ fest konzipierten Aufgabenstellungen lassen sich die zur Beherrschung notwendigen EDV-Kenntnisse minimieren. Je verschiedenartiger und unvorhersehbarer die Aufgabenstellungen und je komplexer die Datenbasis wird, wie wir es überwiegend bei nichtschematischen Aufgaben antreffen, je umfangreicher müssen jedoch die zu erlernenden Systemkenntnisse sein. In den meisten dieser Fälle erweist sich eine derartige Qualifizierung des Nutzers dann jedoch nicht mehr als sinnvoll. Bei dieser Art nichtschematischer Aufgaben empfiehlt es sich deshalb, die indirekte Mensch-Maschine-Kommunikation zu benutzen.

Wenn wir die Prozesse der Leitung und Planung unter dem Aspekt ihrer möglichen Automatisierung betrachten, so gehen wir von der Richtigkeit der folgenden zwei Thesen aus:

1. Auf Grund der Vielschichtigkeit, Komplexität und des überaus hohen Anteils an subjektiver Einflußnahme kann es keine durchgängige Automatisierung der Leitungstätigkeit geben.
2. Auf Grund der unterschiedlichen Charakteristik der Aufgaben der einzelnen Leitungsebenen und ihrer Einflußnahme auf den zu leitenden Leistungsprozeß muß das unterstützende EDV-System je nach Leitungsebene sich in geeigneter Weise darstellen.

Unter dem Aspekt von durchzuführenden Informationsverarbeitungsprozessen lassen sich für Probleme drei Grundklassen angeben:

1. rein semantische Probleme
2. syntaktisch semantisch gemischte Probleme
3. rein syntaktische Probleme /1/

Unter rein semantischen Problemen verstehen wir solche, bei denen die Informationen als Deskriptionen vorliegen und die Methoden der Informationsverarbeitung nicht formalisierbar sind, d.h. der Problembearbeitungs- und Lösungprozeß sind bestimmend getragen von den subjektiven Ansichten des Bearbeiters. Eine mögliche Form der Mensch-Maschine-Kommunikation läßt sich somit nicht zuordnen.

Rein syntaktische Probleme zeichnen sich aus durch das Vorhandensein von Daten und formalisierten Verfahren (Algorithmen) zu ihrer Verarbeitung. Die Aufgabenabarbeitung, wie sie auch genannt wird, unterliegt somit keinen subjektiven Einflüssen des Bearbeiters. Für eine Zuordnung von Kommunikationsformen muß man jedoch weiter unterteilen in die Bearbeitung von schematischen und nichtschematischen Aufgaben. Schematische Aufgaben sollten, wie bereits erwähnt wurde, in einer direkten und nichtschematische in der Mehrzahl in einer indirekten Kommunikationsform abgearbeitet werden. Bei der Mehrzahl der Probleme, die bei den Informationsverarbeitungsprozessen der Leitung und Planung anfallen, tritt jedoch eine Mischung von syntaktischen und semantischen Problemen auf. Diese syntaktisch-semantisch gemischten Probleme werden von P. Keen und M. Morten in dem Buch "Decision Support Systems: an organisational perspective" (Addision Wesley publishing Company 1979) auch als halbstrukturierte Aufgaben bezeichnet. Sie sind gekennzeichnet durch die durch den Menschen zu treffenden Entscheidungen. Wobei im Gegensatz zu den rein semantischen Problemen hier der Mensch den Rechner zielgerichtet zur Bewältigung von Teilaufgaben einsetzt. Der automatisierbare Teil dieser Problemklasse stellt auf Grund seines Charakters fast ausschließlich nichtschematische Aufgaben dar, so daß hier die indirekte Mensch-Maschine-Kommunikation Anwendung finden muß.

Betrachtet man nun die Informationsverarbeitungsprozesse der einzelnen Leitungsebenen einer Leitungshierarchie, so kann grob verallgemeinernd festgestellt werden, daß der Anteil rein syntaktischer Probleme abnimmt und der Anteil rein semantischer Probleme mit steigender Leitungsebene wächst. Diese Tendenz läßt sich dokumentieren, wenn man den Charakter der ablaufenden Informationsverarbeitungsprozesse der einzelnen Leitungsebenen berücksichtigt. Diesem Ziel dient die Abbildung 1. Wobei mit dieser Tabelle kein Anspruch auf Vollständigkeit erhoben wird und man von Verabsolutierungen Abstand nehmen sollte.

| | obere Leitungsebene | untere Leitungsebene |
|---|---|---|
| Bestimmung des Informationsbedarfs | relativ schwer zu ermitteln | da begrenzte Problemstellung relativ gut bestimmbar |
| Erfassungsbereich der Informationen | gesamtbetriebliche Inform. und Inform. zu äußeren Bedingungen | begrenzt auf relativ kleinen Bereich |
| Detailliertheitsgrad | aggregiert | detailliert |
| Aktualität | für die Mehrzahl der Prozesse keine hohen Ansprüche | von großer Bedeutung, da vorrangig operative Tätigkeit |
| Frequenz der Nutzung gleichartiger Aufgaben | gering | häufig |

Die syntaktisch-semantisch gemischten Probleme halten wir in den einzelnen Leitungsebenen zwar vom Inhalt her verschieden, vom Umfang jedoch in etwa gleich. Damit läßt sich als Orientierung aus dem bisherigen schlußfolgern:

1. Auf Grund des Charakters der vorliegenden Aufgaben in der oberen Leitungsebene sollte die Schnittstelle zwischen dem Nutzer und dem AIVS als indirekte Mensch-Maschine-Kommunikation gestaltet werden.
2. Für untere Leitungsebenen sollte in Abhängigkeit von der tatsächlich vorrangig auftretenden Aufgabenklasse sowohl die direkte als auch die indirekte Kommunikation vorgesehen werden.

## 3. Mögliche AIVS zur Bearbeitung von nichtschematischen Aufgaben

Im weiteren Vortrag möchten wir uns nur noch mit dem Teil an Informationsverarbeitungsprozessen beschäftigen, der durch nichtschematische Aufgaben charakterisiert ist. Eine wesentliche Voraussetzung für die Bearbeitung dieser Art von Aufgaben stellen der kurzfristige Zugang zu den gespeicherten Daten und ihre schnelle Auswertung dar. Somit stehen die möglichen Formen und die Verwaltung der Datenbasis sowie die Möglicheiten ihrer Auswertung im Mittelpunkt. Bei den Formen von Datenbasen möchten wir uns jedoch auf die einheitlicher Datenbasen beschränken, da nur sie die Grundlage bilden können für eine erfolgreiche Bewältigung der meist komplexen nichtschematischen Aufgaben. Unter einer einheitlichen Datenbasis verstehen wir die datentechnische "Zusammenfassung von verschiedenen Datenbeständen (Dateien) eines abgeschlossenen Organisationsbereiches" /2/, wobei gleiche Merkmale gleich aufgebaut und die Schlüssel einheitlich sind.

Somit lassen sich folgende zu einheitlichen Datenbasen zählen:

1. Projektdateibasis: Das sind Datenbasen, die zum großen Teil aus den Stammdateien der einzelnen EDV-Projekte eines AIVS stammen, zentral, aber auch dezentral verwaltet werden und der Forderung nach Einheitlichkeit der Merkmale und der Verschlüsselung genügen.
2. Potentielle Datenbasis: Das sind Datenbasen, die sich gründen auf vorhandene Dateien, die durchaus nicht einheitlich hinsichtlich der Verschlüsselung und der Merkmale zu sein brauchen, wobei jedoch entsprechende Schlüssel- und Merkmalkonvertierungsprogramme existieren. Mit diesen Programmen wird erst bei Bedarf die einheitliche Datenbasis für eine gewisse Zeit erstellt.
3. Datenbank: EineDatenbank ist "in ihrem Wesen nach ein System, das aus redundanzfrei gespeicherten Daten einschließlich der gespeicherten Darstellung der Beziehungen und Abhängigkeiten zwischen den Daten besteht und den direkten Zugriff zu den Daten ermöglicht." /3/

Die Verwaltung von Datenbasen stellt im wesentlichen ein organisatorisches Problem dar. Daß es heute dafür bereits weitgehende Möglichkeiten der Softwareunterstützung gibt, zeigen die Serviceleistungen der existierenden Datenbankbetriebssysteme. Da sie im allgemeinen bekannt sind, möchten wir in unserem Vortrag auf Erläuterungen verzichten. Neben dieser Art halten wir in bestimmten Anwendungsfällen, auf die später noch eingegangen werden soll, die Möglichkeit der Verwaltung der Datenbasis durch ein System von organisatorischen Regelungen für möglich. Darunter sind die folgenden, durch die schon erwähnte Informationszentrale zu führenden Arbeitsmittel zu verstehen:

- Datenkatalog
- Programmkatalog
- Listenkatalog
- Verzeichnis der Dateibeschreibungen
- Verzeichnis der Nutzungsbeschreibungen
- Schlüsselkatalog
- Dateiverzeichnis

Für die Auswertung der Datenbasis gibt es verständlicherweise die vielfältigsten Möglichkeiten wie:

- Dialogsprachen
- Listengeneratoren
- Programmgeneratoren
- Programmiersysteme u.ä.

Bei Datenbanksystemen macht dieser Teil der Auswertung das oder die externen Schemata aus. Ein oder mehrere systemeigene oder selbständige Programmiersprachen werden zur Auswertung der Daten benutzt.

Eine andere Möglichkeit sind Datenauswertungssysteme.
Unter einem Datenauswertungssystem (DAS) verstehen wir ein System von Softwareprodukten mit möglichst hohem Automatisierungsgrad, das zur Lösung nichtschematischer Aufgaben durch Auswertung von ein oder mehreren Dateien aus Projektdateibasen oder aus potentiellen Datenbasen eingesetzt wird. Die Art der eingesetzten Softwareerzeugnisse in einem DAS ist abhängig von:

- dem konkreten Aufbau der Datenbasis
- der jeweiligen Aufgabencharakteristik und dem vorhandenen Nutzerprofil
- der Hardware und dem Betriebssystem

Für die Aufgaben aus dem Prozeß der Leitung und Planung werden dies im wesentlichen Listen- und Programmgeneratoren sein.

Aus dem bisher Dargestellten lassen sich zwei Systeme zur Bearbeitung von nichtschematischen Aufgaben erkennen:

1. Datenbanksysteme: Wobei wir darunter eine Datenbank, ein Datenbankbetriebssystem, ein Exekutivsystem zur Auswertung der Daten und einen Datenbankadministrator zur Organisation und Überwachung der Funktionen verstehen.
2. Datenauswertungs- und Verwaltungssysteme (DAVS): Ein DAVS umfaßt eine einheitliche Datenbasis, für die vorgefertigte Programme und organisatorische Regelungen zu ihrer Verwaltung existieren, ein Datenauswertungssystem zu ihrer Auswertung und eine Informationszentrale.

## 4. Einsatzbedingungen für DBS und DAVS

Es ist nicht das Ziel dieses Abschnittes, DBS und DAVS in ihrer Leistungsfähigkeit gegenüberzustellen oder eine Wertung vorzunehmen. Wir wollen vielmehr versuchen, allgemeine Bedingungen für den Einsatz von DBS und DAVS zu formulieren.
Gegenüber der herkömmlichen Projektierung und Programmierung werden für DBS im wesentlichen die folgenden Vorteile gesehen:

- Die Datenbank bietet größere Möglichkeiten der Verarbeitung und Verdichtung der Daten, d.h. durch die funktions- und aufgabenunabhängige Speicherung der Daten ist eine größere Zahl von Fragestellungen mit vertretbarem Aufwand möglich.
- Die Datenbank gewährleistet eine größere Aktualität des Gesamtdatenbestandes.
- Bei einer Datenbank sind weniger Kommunikationsprozesse notwendig, um Daten aufzufinden. Es wird der direkte Zugriff des Nutzers zum gesamten Datenbestand ermöglicht.

Unterstellen wir diese Vorteile eines DBS, wobei in der Literatur weitere angeführt werden, so müssen jedoch auch einige Schwierigkeiten beim Aufbau von DBS erwähnt werden.

- Nicht alle DBS genügen in gleicher Weise für jeden Anwendungsfall diesen Vorteilen. Deshalb ist eine auf den konkreten Anwender zugeschnittene Auswahl eines geeignetenDBS notwendig, auf Grund der in der DDR begrenzten Anzahl jedoch recht schwierig.
- In der Mehrzahl der Fälle sind die Einrichtungen, bei denen ein DBS eingeführt werden soll, keine Neulinge auf dem Gebiet der Anwendung der EDV. Es ist hier deshalb notwendig, wenn nicht sogar teilweise in den Anwenderbereichen, so auf jeden Fall im Rechenzentrum oder in der entsprechenden Einrichtung, die bestehende Organisation der Datenerfassung, -prüfung, -einspeicherung und -auswertung zu ändern. Eine solche Umstellung ist erfahrungsgemäß mit einer Reihe von Problemen verbunden.
- Datenbankbetriebssysteme sind in der Regel umfangreiche Programmsysteme. Da diese verständlicherweise nicht völlig fehlerfrei arbeiten, treten auch aus programmtechnischer Sicht Anfangsschwierigkeiten auf.
- Durch den Umfang der Datenbankbetriebssysteme und die anzunehmende Häufigkeit der Nutzung der Datenbank werden erhöhte Anforderungen an die jeweils genutzte Hardware gestellt. Forderungen zum einen nach einer umfangreichen Peripherie, insbesondere an Direktzugriffsdatenträger, und zum anderen nach einem über längere Zeit stabilen Lauf der Rechenanlage.

Die aufgezeigten Schwierigkeiten machen deutlich, daß es nicht in jedem Fall sinnvoll ist, DBS einzusetzen. Die nachfolgenden Punkte sollen dem Anwender eine Unterstützung bei der Entscheidung sein.

1. Haben die Daten tatsächlich einen solchen Umfang erreicht, daß sie nur noch durch ein DBS verwaltet werden können? Ist die Verflechtung der Daten so stark, daß ihre Struktur nur noch in einer Datenbank widergespiegelt werden kann?
3. Eine Senkung der Redundanz imDatenbestand, insbesondere bei der Primärdatenerfassung, läßt sich durch gute organisatorische Maßnahmen ebenfalls ermöglichen, wobei die Redundanzarmut eines DBS jedoch nicht erreicht wird.

4. DBS sind gekennzeichnet durch eine hohe Aktualität des gesamten Datenbestandes. Ist diese Aktualität für alle Daten vonnöten, oder ließe sie sich auch für die geforderten Daten durch gesonderte Programme erreichen?
5. Die Möglichkeiten der ONLINE-Verarbeitung sind in großem Stil wohl nur durch DBS realisierbar. Zu fragen ist, ob diese Art der Verarbeitung für den Informationsendverbraucher tatsächlich von Nutzen ist.

Auch wenn die letztgenannten Punkte den Anschein geben, kann es nicht Anliegen sein, gegen die Einführung von DBS zu argumentieren.

Ziel unseres Beitrages war es, eine Einordnung von möglichen AIVS zur Bearbeitung von nichtschematischen Aufgaben aus dem Leitungsprozeß vorzunehmen und sowohl die Eignung von DAVS als auch DBS für diese Art von Aufgaben hervorzuheben.

## Literaturhinweise

/1/ K. Fuchs-Kittowski, H. Kaiser, R. Tschirschwitz, B. Wenzlaff: "Informatik und Automatisierung", Berlin 1976

/2/ Lutz, Theo: "Adressketten und Datenbanktechnik", BTA & BTO 4/1968, S. 178

/3/ Wagner, Klaus: "Gestaltungs- und Entwicklungsziele künftiger Systemunterlagen für Datenbanken - Grundfragen ihrer Realisierbarkeit, Diss. Berlin 1981

## Verfasser

Dipl.-Math. Manfred Hagen
Dipl.-Ing. Egbert Klaembt
Dipl.-Wirt. Peter Schirmbacher
Humboldt-Universität zu Berlin
Organisations- und Rechenzentrum
1086 Berlin
Unter den Linden 6

Fichtner, Norbert

## Interdisziplinäre Zusammenarbeit bei Anwendung der automatisierten Informationsverarbeitung im Gesundheitswesen und in der Medizin

Die Notwendigkeit der interdisziplinären Zusammenarbeit bei Anwendung der automatisierten Informationsverarbeitung im Gesundheitswesen und in der Medizin ergibt sich aus

- den gesundheitspolitischen und technisch-organisatorischen Erfordernissen bei der Überwachung und weiteren Verbesserung des Gesundheitszustandes der Bevölkerung
- dem progressiven Anwachsen des internationalen Erkenntnisstandes in der Medizin
- der wissenschaftlich-technischen Fortschritt in Einheit mit dem Wachstum der Produktivkräfte
- der gesellschaftlichen Forderung nach effektiverer Auslastung der dem Gesundheits- und Sozialwesen zur Verfügung gestellten finanziellen, materiellen und personellen Fonds.

Diese objektiv vorhandenen Bedingungen äußern sich in einer ständigen Zunahme von Informationen, die es zu verarbeiten gilt, um die Errungenschaften der modernen Medizin gezielt zum Wohle der Bürger nutzbar zu machen sowie die Qualität und Wirksamkeit der medizinischen Betreuung kontinuierlich zu erhöhen.

Partei und Staatsführung nehmen seit Jahren auf diese Entwicklung gezielt Einfluß und im Ergebnis dieser Bemühungen konnte auf dem X. Parteitag der SED festgestellt werden, daß die Forderung nach engem Zusammenwirken von Grundlagenforschung, angewandter Forschung sowie technischer und technologischer Forschung und Entwicklung immer besser verwirklicht wird. /1/
Weiterhin wird festgestellt:
"Die enge Beziehung der Forschungskollektive zum wissenschaftlichen Gerätebau und zur Entwicklung hochempfindlicher Technik ist zu einem wichtigen Bestandteil des Forschungsprozesses geworden. Dazu wird auch die Ausarbeitung und Bereitstellung leistungsfähiger Verfahren und Systeme der Datenverarbeitung und -übertragung

sowie der Ausbau der wissenschaftlichen Information und Dokumentation." /1/

Die Anwendung der automatisierten Informationsverarbeitung im Gesundheitswesen und in der Medizin kann in folgende Komplexe grob untergliedert werden:

- Unterstützung der Prozesse der Leitung und Planung des GSW
- automatisierte Meßwertverarbeitung
- rechnergestützte patientenbezogene und betriebliche Informations- und Dokumentationssysteme
- Unterstützung der Dispensairebetreuung
- Unterstützung der medizinischen Forschung
- Rationalisierung der Versorgungsprozesse.

Die Vielfalt der informationellen Beziehungen auf diesen Gebieten verdeutlicht, daß die automatisierte Informationsverarbeitung direkt und indirekt in die gesundheitliche Betreuung der Bürger eingreift. Gleichermaßen geht aus den zahlreichen Einsatzfällen in den genannten Bereichen hervor, daß Fachwissenschaftler der 3 unterschiedlichen Disziplinen eng mit dem Mediziner zusammenarbeiten müssen, um die anfallenden Informationen zu beherrschen, zu verarbeiten und zu nutzen. Die hierbei beteiligten Wissenschaftsdisziplinen lassen sich grob in drei Gruppen gliedern. Das sind:

1. Durchgängig Mediziner der verschiedenen Fachrichtungen
2. Durchgängig Informationswissenschaftler und -techniker
3. Wissenschaftler und Fachleute der unterschiedlichen Wissenschaftsdisziplinen entsprechend der Spezifik der genannten Einsatzgebiete.

Die interdisziplinäre Zusammenarbeit vollzieht sich zwangsläufig und selbstverständlich im Prozeß der Arbeit. Andererseits ist sie jedoch verstärkt und bewußt durchzusetzen und zu qualifizieren, um die Effektivität der Arbeit zu erhöhen, die medizinische Betreuung weiter zu verbessern und die Wirksamkeit der Forschung zu erhöhen. Eben dieses Anliegen wird auch durch staatliche Maßnahmen unterstützt, so durch die Einführung des postgradualen Studiums für naturwissenschaftliche und technische Hochschulkader

sowie Dipl.-Psychologen und Dipl.-Soziologen im Gesundheitswesen 1981. Die Weiterbildung der naturwissenschaftlichen und technischen Hochschulkader erfolgt gegenwärtig in 18 Fachrichtungen, u.a. auf den Gebieten Biochemie, Biomathematik, Biophysik, klinische Chemie und Labordiagnostik, klinische Strahlenphysik, Kommunalhygiene, klinische Arbeitshygiene und medizinische Soziologie.

Ziel des postgradualen Studiums ist es, in enger Gemeinschaftsarbeit von naturwissenschaftlichen und technischen Hochschulkadern mit Ärzten, die Medizin durch die Nutzung neuer Erkenntnisse der Biologie, Chemie, Mathematik, Physik, Technik, Psychologie und Soziologie zu bereichern, indem es die genannten Hochschulkader befähigt, berufsbezogene spezielle Tätigkeiten in der medizinischen Forschung und Betreuung in höher Qualität auszuüben./2/

Ebenso wie sich der Informationswissenschaftler und -techniker in die Spezifik des medizinischen Fachgebietes einzuarbeiten hat, müssen auch an den Mediziner, der Verfahren und Methoden der automatisierter Informationsverarbeitung nutzt, bestimmte Bildungsforderungen als Grundlage einer effektiven interdisziplinären Zusammenarbeit gestellt werden. Das sind u.a. Kenntnisse über die Leistungsfähigkeiten, die Möglichkeiten und Grenzen der Nutzung eine elektronischen Datenverarbeitungsanlage mit Schwerpunkten wie:

- Wer ist Anlaufpartner im Rechenzentrum für den Mediziner?
- Welche vorhandenen Möglichkeiten können genutzt werden?
- Abschätzung des Nutzens und Aufwandes maschineller Aufbereitungen
- Kenntnis über die Anforderungen, die an das von dem Mediziner übergebene Material zu stellen sind usw.

Einige Bemerkungen zu konkreten Anwendungsgebieten. Bei der rechnergestützten Bereitstellung von Informationen zur Rationalisierung und Qualifizierung der Leitungs- und Planungsprozesse im Gesundheits- und Sozialwesen gehören zur Realisierung der interdisziplinären Zusammenarbeit zu der o.a. 3. Gruppe der beteiligten Fachleute Gesundheitspolitiker, Staatswissenschaftler, Ökonomen und Juristen.
Zur materiell-technischen Absicherung der ständigen Erhöhung von

Qualität und Wirksamkeit der medizinischen Betreuung der Bürger bei der Prophylaxe, Diagnostik, Therapie und Metaphylaxe wurden bis 1980 mehr als doppelt so viele Investitionen wie im vorhergegangenen 5-Jahrplan dem Gesundheitswesen zur Verfügung gestellt. Insgesamt wurden im Zeitraum von 1970 bis 1980 ca. 4 Milliarden Mark dem GSW zur Verfügung gestellt. Die Zahl der Beschäftigten im GSW beträgt annähernd ½ Million; das entspricht 5,7 % des gesamten gesellschaftlichen Arbeitsvermögens der DDR. Mehr als 42 000 Ärzte und Zahnärzte sind in der DDR tätig, jährlich sind etwa 2,3 Millionen Krankenhausabgänge sowie eine ständig steigende Anzahl von Konsultationen im ambulanten Bereich organisatorisch zu bewältigen. Die Anzahl der Konsultationen in staatlichen ambulanten Einrichtungen einschließlich der Hausbesuche betrug im Jahre 1960 noch 61,7 Millionen; 1980 waren es 142,3 Millionen. Zur Zeit sind in der ambulanten Betreuung im Wohngebiet von einem Arzt (im Mittel aller Fachrichtungen) je Arbeitstag etwa 50 Konsultationen und Hausbesuche zu bewältigen, wenn 200 effektive Arbeitstage/Jahr angenommen werden.
Die interdisziplinäre Zusammenarbeit bei der rechnergestützten Verarbeitung der sich aus den genannten Kennziffern ergebenden Vielzahl von Informationen muß sich aus gesundheitspolitischer und volkswirtschaftlicher Sicht konzentrieren auf:

- die Ermittlung des Bedarfes der Bevölkerung an medizinischer und sozialer Betreuung
- die Beurteilung der Bedarfsbefriedigung
- die Einschätzung der Qualität der medizinischen und sozialen Betreuung
- die Verbesserung der Ausnutzung der Fonds und der Auslastung der Kapazitäten
- die Nutzung von Informationen aus der epidemiologischen Forschung durch ihre Verknüpfung mit Leitungs- und Planungsinformationen.

Die interdisziplinäre Zusammenarbeit bei der rechnergestützten Lösung dieser Aufgabenstellungen beginnt bereits bei der Analyse der nutzbaren bzw. aufzubauenden Datenfonds und setzt sich bei der Entwicklung der konkreten EDV-Projekte entsprechend den Stufen E1 bis E6 der Nomenklatur der Arbeitsstufen und Leistungen von Aufgaben des Planes Wissenschaft und Technik fort. Dabei

wird der Umfang der von anderen Fachdisziplinen außerhalb der Informationsverarbeitung zu erbringenden Leistungen mit den einzelnen Projektierungsstufen geringer.

Die automatisierte Meßwertverarbeitung dürfte das Einsatzgebiet der elektronischen Datenverarbeitung sein, auf dem die interdisziplinäre Zusammenarbeit am ausgeprägtesten ist und den größten Umfang erreicht. Die hier beabsichtigte rechnergestützte Entscheidungsfindung für Diagnostik, Therapie und Prognose erstreckt sich auf die Gebiete:

- Laborautomatisierung
  (z.B. Hämathologisches Labor, klinisch-chemisches und mikrobiologisches Labor, Isotopenlabor)
- Radiologie
- Funktionsdiagnostik/Biosignalanalyse
- Patientenüberwachung in der Intensivtherapie
- Bildverarbeitung.

Gerade auf diesem Anwendungsgebiet der automatisierten Meßwertverarbeitung tangiert die interdisziplinäre Zusammenarbeit in gravierender Weise die Industrie. Sie hat entscheidenden Anteil an der Entwicklung und der Bereitstellung von Systemlösungen, die in breitem Umfang im GSW einsetzbar sind. Als Beispiel hierfür sei das Mikrorechnersystem zur Bestrahlungsplanung Robotron dopsy-r angeführt. Auf der Grundlage des Mikrorechners Robotron K 1630 und einer Konfiguration, die den speziellen Erfordernissen der Nutzer angepaßt werden kann, wurde die notwendige Software gemeinsam von Physikern und Medizinern des Zentralinstitutes für Krebsforschung und Mitarbeitern des Rechenzentrums des Zentralinstitutes für Molekularbiologie der Akademie der Wissenschaften der DDR entwickelt.

Die Bemühungen progressiver Mediziner und aufgeschlossener, kooperationsbereiter Naturwissenschaftler und Techniker unterstützten und unterstützen, beschleunigten und beschleunigen die Entwicklung der Medizin zu einer analytischen und weitgehend quantitativ objektivierbaren Wissenschaft in dem Bewußtsein, daß diese in der historischen Anlage bedingte empirische Wissenschaft in starkem Maße von der Entwicklung der Technik und den entsprechenden mathematisch-naturwissenschaftlichen Disziplinen beein-

flußt wird. Entsprechend der engen und wechselseitigen Verbindung zwischen deskriptiver und analytischer Methodik in der medizinischen Forschung kommt dabei gerade der mathematischen Statistik eine spezifische Bedeutung bei der Gewinnung und Absicherung quantifizierbarer medizinischer Ergebnisse zu. Hierbei erscheint es als wesentlich, unsere Mitarbeiter zu motivieren, die schnelle, zuverlässige und umfassende Erhebung und Auswertung der medizinischen Datenbestände zu unterstützen. Die Kompliziertheit der angewendeten mathematischen Modelle und Verfahren, die Notwendigkeit, ständig und schnell zu Zwischenergebnissen zu gelangen, verschiedene Möglichkeiten zu testen, korrigierte Rechnungen durchzuführen erfordern die verständnisvolle, umfassende, interessierte und wissenschaftlich verantwortungsbewußte problemorientierte Mitarbeit und Zusammenarbeit der Fachwissenschaftler in der Medizin und in den Rechenzentren. Es ist ein gesellschaftliches Erfordernis, einen aufwandsadäquaten Nutzen beim Einsatz der EDV im Interesse unserer Bürger zu erbringen.

Wenn ich eingangs gesagt habe, daß die umfassende, effektive und sachgerechte Anwendung mathematisch-statistischer Methoden und Verfahren sowie die Nutzung entsprechender mathematischer Modelle in der medizinischen Betreuung und Forschung den Einsatz der elektronischen Datenverarbeitung erfordern, leitet sich gleichzeitig aus den Erfordernissen der modernen praxiswirksamen medizinischen Forschung die spezifische Aufgabenstellung der mathematisch und informationstechnisch orientierten Disziplinen ab. Bei der epidemiologischen und medizinsoziologischen Forschung wird im allgemeinen pro Person eine bestimmte Anzahl von Merkmalen in Form von statistischen Variablen erfaßt, wobei die Auswertung der Datensätze sequentiell erfolgt. Aus der Vielzahl und Kompliziertheit der zu berücksichtigenden Merkmale und ihrer Ausprägungen ergeben sich bei der Anwendung mathematisch-statistischer Verfahren unter Nutzung der EDV häufig Schwierigkeiten. Sie zu überwinden, stellt ebenso eine Aufgabe des in Anspruch genommenen Rechenzentrums dar, wie die Mithilfe und methodische Beratung bei der Lösung von - teilweise subjektiv bedingten - Problemen. Das betrifft vor allem die Unterstützung bei der:

- Herausarbeitung klarer und arbeitsfähiger Zielstellungen für die statistische Auswertung der Daten sowie bei Unterstützung

einer exakten Aufgabenformulierung
- Erarbeitung methodischer Konzeptionen bezüglich der Datenerhebung und ihrer weiteren Verarbeitung
- Erfassung eindeutiger im Sinne mathematischer Algorithmen verarbeitungsfähiger Daten
- Erarbeitung und methodisch übersichtliches Angebot einer ausreichenden Anzahl in ihrer Anwendung und Ergebnisdarstellung getesteter mathematischer Modelle und Verfahren.

Die objektive Notwendigkeit einer engen und über die Kooperation hinausgehenden Zusammenarbeit zwischen Medizinern und Wissenschaftlern informationstechnisch und mathematisch orientierter Disziplinen liegt im vergleichsweise geringen Integrationsgrad mathematischer Methoden und Verfahren in der Medizin begründet. Die Biometrie steht als eigenständiges Fachgebiet neben den biologischen Wissenschaften, während in anderen Wissenschaftsdisziplinen wie Physik, technische Mechanik, Elektrotechnik, Ökonomie u.a. die Integration der Mathematik in der Fachwissenschaft selbst vollzogen ist. Dadurch erhält der in volkswirtschaftlichen und wissenschaftlich-technischen Bereichen tätige Informationswissenschaftler die Problemstellung von der fachlichen Seite her soweit aufbereitet, daß die mathematische Algorithmierung bzw. die problemanalytische, programmier- und rechentechnische Bearbeitung relativ problemlos erfolgen kann. Die dem Rechenzentrum übergebene fachspezifische Problemstellung ist in ihrer Fragestellung weitgehend durchdacht und relativ ausgereift.

Die Befähigung des Mediziners zur Nutzung der automatisierten Informationsverarbeitung als wesentlicher Faktor zur integrativen Zusammenwirkung von theoretischer und klinischer Medizin in Verbindung mit technisch-naturwissenschaftlichen Disziplinen sollte durch die praxisnahe Einbeziehung des Fachgebietes automatisierte Informationsverarbeitung/mathematische Statistik in die Aus- und Weiterbildung der medizinischen Kader erreicht werden. Hierbei haben wir Erfolge erzielt; aber wir haben hier noch erhebliche Arbeit zu leisten, gemeinsame Arbeit zu leisten, um auch diese Möglichkeiten der Weiterbildung bewußt auszunutzen, die integrative Zusammenarbeit zu stärken und damit die Forschungs- und Betreuungsprozesse in der Medizin zu unterstützen. Dabei sollten bei der Weiterbildung der medizinischen Kader die

funktionellen Aspekte der Datenverarbeitung hervorgehoben werden, während die technischen Aspekte weitgehend eingeschränkt werden sollten.

Die Darlegungen sind als Anregung und Unterstützung gedacht, die Forderungen von Partei- und Staatsführung nach engem Zusammenwirken aller Wissenschaften, aller Wissenschaftsdisziplinen und Ausprägung der sozialistischen Gemeinschaftsarbeit zur Stimulierung der wissenschaftlichen Arbeit zu erfüllen und der herausgestellten Bedeutung der Entwicklung der Grundlagen- und angewandten Forschung in den Biowissenschaften einschließlich den naturwissenschaftlichen Grundlagen der Medizin gerecht zu werden. Hierbei tragen wir als Informationstechniker und -theoretiker eine spezifische Verantwortung.

Literaturhinweise

/1/ Bericht des Zentralkomitees der Sozialistischen Einheitspartei Deutschlands an den X. Parteitag der SED
Dietz Verlag Berlin 1981

/2/ Anweisung über das postgraduale Studium für naturwissenschaftliche und technische Hochschulkader sowie Diplompsychologen und Diplomsoziologen im GW vom 01.04.1981
Verfügungen und Mitteilungen des MfGe Nr.4 vom 04.05.1981

Verfasser:

Dr.Dr.sc.nat. Norbert Fichtner
Institut für Sozialhygiene und Organisation des
Gesundheitsschutzes "Maxim Zetkin"
Organisations- und Rechenzentrum

1134 Berlin, Nöldnerstr. 34-36

Melzer, Jürgen

# Die Beziehung zwischen Struktur und Auswertung bei patientenbezogenen Datenbeständen

## 1. Einleitung

Die Vermeidung von Datenfriedhöfen, d.h. die Existenz von kaum oder schwer auswertbaren Datenbeständen innerhalb von Informationssystemen, stellt heute ein anerkanntes Problem dar. Besonders in der Medizin, wo mehrjährige retrospektive Auswertungen sehr oft Ausgangspunkt bzw. wesentlicher Bestandteil von Forschungsprozessen sind, erweist sich die Berücksichtigung des Zusammenhangs von Gestaltung, Struktur und Auswertbarkeit als besonders notwendig.
Betrachtet man die Datenauswertung in ihrer Gesamtheit, d.h. einschließlich aller notwendigen Zwischenschritte und Hilfsarbeiten, wird deutlich, daß es sich in struktureller Hinsicht um mehrere Ebenen handelt:

- reale Struktur der Wirklichkeit
- logische Struktur der Daten
- physische Struktur der Daten.

Die Auswertung ist damit auch ein Transformationsproblem (Übergang zwischen den Ebenen), das nach Auffassung des Autors nicht vollständig automatisierbar ist. D.h. im allgemeinen ist die Lösung eines Auswertungsproblems immer mit einem Anteil intelligenter menschlicher Tätigkeit verbunden. Hierin liegt die wesentlichste auswertungsspezifische Aufgabe der Informationszentrale (siehe auch Beitrag von Hagen, Klaembt, Schirmbacher).

## 2. Begriffserläuterungen

Die folgenden Erläuterungen resultieren aus der Sicht und dem Verständnis des Autors. Es wird damit keinesfalls beabsichtigt, eine über das Thema hinausgehende allgemeingültige Definition zu geben.

Struktur der Daten (physische):

- Begriff für das verwendete Prinzip der Anordnung einer niederen in einer höheren Organisationseinheit der Daten (z.B.

Datum im Datensatz, Datensatz in der Datei)
- sichert die Ordnung und damit die Identifikation (Adressierung) als Voraussetzung einer zweckmäßigen Verfügbarkeit der Daten
- nach ihr erfolgt die jeweils konkrete Datenorganisation.

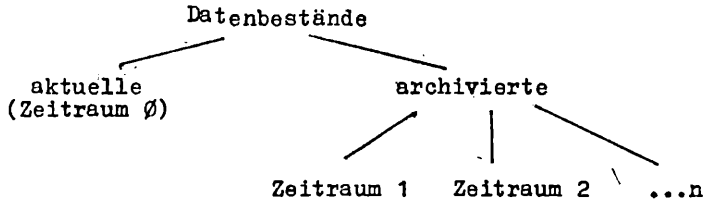
Gestaltung der Daten:
Umfaßt
- Problematik des Umfangs, der Auswahl und der inhaltlichen Beziehung der erfaßten (gespeicherten) Daten sowie
- Problematik der Abbildungstreue der Daten in Bezug auf den abzubildenden Prozeß.

Logische Struktur der Daten:
- Bildet die Grundlage zur Übersicht über die Datenmenge (Orientierungsmittel)
- ermöglicht die Identifikation der Daten
- ist die Voraussetzung für jegliche Nutzbarmachung von Datenmengen
- entsteht auf der Basis der Gestaltung der Daten.

Auswertung (Datenauswertung im Sinne des Beitrags):
Bezeichnet den diskontinuierlichen Prozeß der Lösung bzw. Bearbeitung von vorwiegend einmaligen Informationsverarbeitungsaufgaben mit Hilfe der automatisierten Informationsverarbeitung auf der Basis bereits vorliegender maschinell gespeicherter Daten.
Dabei bedeutet
"vorwiegend einmalig": Es gibt Aufgaben (z.B. jährliche Analysen), die im bestimmten zeitlichen Rhythmus wiederholt werden.
"mit Hilfe der automatisierten Informationsverarbeitung": Die Lösung umfaßt menschliche und maschinelle Tätigkeiten.
" auf der Basis gespeicherter Daten": Die Erfassung und Aufbereitung (Prüfung) ist nicht Bestandteil der Datenauswertung.
Bei vergleichenden Auswertungen - zu dieser Gruppe gehören u.a. die in einem bestimmten zeitlichen Rhythmus sich wiederholenden Auswertungsaufgaben und natürlich auch die bereits erwähnten,

für die medizinische Forschung besonders wichtigen, mehrjährigen retrospektiven Auswertungen - ist die Forderung nach einer kontinuierlichen Qualität der Auswertung hinsichtlich der Brauchbarkeit der Ergebnisse von entscheidender Bedeutung. Im übertragenen Sinne könnte man auch von einer "Stetigkeit der Auswertung" sprechen.

Zum Verständnis:




Läuft eine Auswertung über mehrere inhaltlich vergleichbare Datenbestände (z.B. über mehrere Zeiträume) so muß zur Wahrung der Forderung nach einer kontinuierlichen Qualität (bzw. Stetigkeit) der Auswertung für jede Auswertungsaufgabe (AwA) gelten:

> Die Existenz einer Lösung auf der Basis eines Teiles des auszuwertenden Datenbestandes sichert eine vergleichbare Lösung auf der Basis jedes beliebigen Teiles des auszuwertenden Datenbestandes.

## 3. Die Dynamik des Datenbestandes als Problem der Struktur

Für komplexe rechnergestützte Informationssysteme (IS) jeglicher Art gilt, daß sie im allgemeinen nicht mit einem Schlag im vollen geplanten Umfang eingeführt werden können. Die Gründe hierfür sollen hier nicht näher erörtert werden. Nur soviel sei gesagt, daß sie sehr vielfältig und nicht nur von trivialer Art sind. Neben technischen und organisatorischen Problemen treten mit wachsendem Umfang des IS vor allem soziale Probleme auf.

Besonders in den Betreuungseinrichtungen des Gesundheitswesens, wie z.B. in den Versorgungskrankenhäusern mit ihren historisch gewachsenen Organisationsstrukturen und Leistungshierarchien, ist die Akzeptanzproblematik eine nicht zu unterschätzende Größe. Die schrittweise Einführung des IS ist hier gewöhnlich der einzige gangbare Weg. Die Maßnahme führt natürlich dazu,

daß der Umfang der vom IS erfaßten, zu verarbeitenden und zu speichernden Informationen bzw. Daten sich stufenweise erweitert.
Diese Diskontinuität hinsichtlich des Umfangs und der inhaltlichen Gestaltung der für ein IS relevanten Daten ist aber keinesfalls mit der vollen Einführung des IS beseitigt. IS realisieren ja immer in irgendeiner Form eine Abbildungsfunktion der realen Umwelt, in der sie eingebettet sind. IS sind ein Teil der Betriebs- bzw. Krankenhausorganisation. Oft sind sie auch noch in regionale Organisationen (Signierstreifen, Berichtswesen) eingebunden. Damit unterliegen sie mehr oder weniger einem ständigen Druck nach Anpassung d.h. Veränderung.
Als Beispiel ~~sollte~~ für solche äußere Einflüsse, die in den letzten Jahren auf Patienteninformationssysteme unabhängig von ihrem Einsatzort und ihrer Entwicklung wirkten, seien genannt:

- Änderung des standardisierten Krankenblattes (Aufbau)
- Änderung des Signierstreifens (Aufbau und Inhalt, Hinzunahme bisher nicht erfaßter Informationen)
- Einführung der 9. Revision des WHO-Diagnose-Schlüsselverzeichnisses (veränderte Schlüssel, Wegfall des Sicherheitsgrades).

Innerhalb unseres Krankehauses wurde eine Umbenennung der Stationen vorgenommen (neben neuen Bezeichnungen wurden Beziehungen unter den Daten dadurch verändert).
Bei Patienteninformationssystemen, die im wesentlichen Informationen zum medizinischen Betreuungsprozeß der einzelnen Patienten verarbeiten, folgt insbesondere aus der ständigen Weiterentwicklung der medizinischen Wissenschaften, z.B. durch neue Diagnostikverfahren, neue therapeutische Erkenntnisse (u.a. auch als Folge der Auswertung von entsprechenden patientenbezogenen Daten, dh. infolge der eigenen Arbeit), neue Medikamente ua.m., die Notwendigkeit einer ständigen Anpassung der vom IS zu verarbeitenden Informationen an den abzubildenden Prozeß. Neben diesen objektiven Gründen treten noch subjektive Faktoren als Ursache für Änderungen auf. Hier wären insbesondere die persönlichen Interessen und Beziehungen der am Prozeß beteiligten oder ihn leitenden Kadern zu nennen. Die historisch gewachsene Organisation und Hierarchie des medizinischen Betreuungsprozesses begünstigt das Auftreten und Wirksamwerden dieser

subjektiven Faktoren.

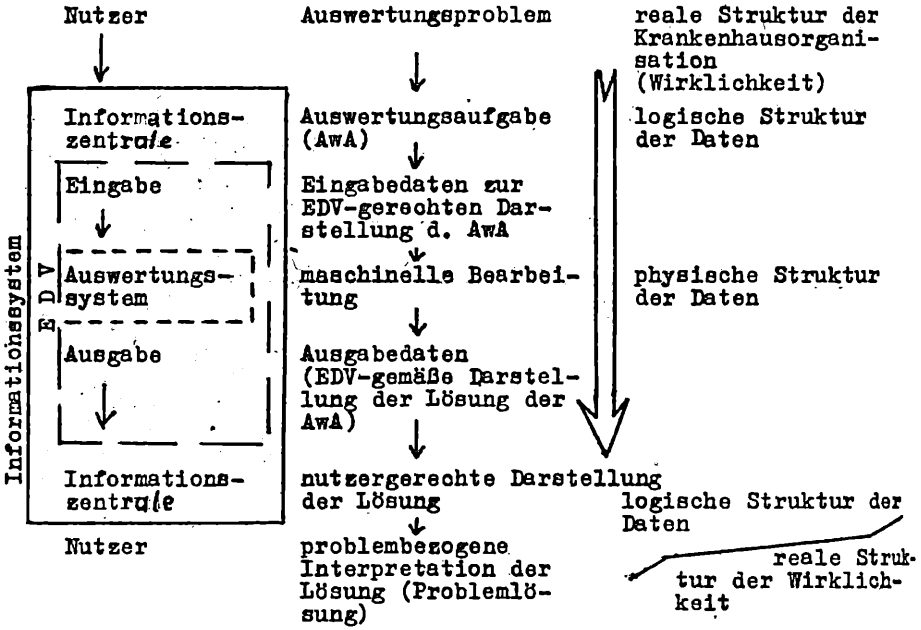
Die Einordnung des IS und die jeweilige, eine Veränderung des Datenbestandes verursachende, Folge:

| Fakt | Folge |
|---|---|
| Die Komplexität des IS<br>- organisatorische<br>- soziale<br>- technische<br>- programmseitige | Die schrittweise Einführung des IS |
| Die Einbettung des IS in Organisationen<br>- betriebliche<br>regionale<br>- landesweite | Anpassung an veränderte Organi-Organisationen |
| Die Abbildungsfunktion des IS<br>- Prozesse<br>- Objekte | Anpassung an die Veränderungen der abzubildenden Objekte bzw. Prozesse |
| Das IS als Realisierung von Datenverarbeitungstechnologie<br>- Hardware<br>- Software | Berücksichtigung der anhaltenden Evolution auf diesem Gebiet |

Fazit:
Durch die Einbettung der IS in gesellschaftliche Organisationen und die Abbildung realer gesellschaftlicher Prozesse durch das IS unterliegt die zu verarbeitende Informationsmenge einer ständigen Dynamik, d.h. die dazugehörigen Datenbestände können nur eine relative Stabilität besitzen, wenn das IS seiner Abbildungsfunktion und damit seiner Aufgabe gerecht werden soll. Daraus folgt zwangsläufig, die Struktur der zur Auswertung vorgesehenen Daten (aktuelle wie archivierte) muß eine ständige inhaltliche Anpassung ermöglichen und gleichzeitig die Voraussetzung dafür bieten, daß die dabei entstehenden ihrer Gestaltung nach verschiedenen Teile des Datenbestandes bei geeigneten Auswertungssystemen die Forderung nach einer kontinuierlichen Qualität der Auswertung erfüllen.

## 4. Die Auswertung als strukturelles Mehrebenenproblem

Es existieren noch weitere Beziehungen zwischen Struktur und Auswertung. Das wird deutlich durch das Wechselspiel zwischen Nutzer und Informationssystem im Zusammenhang mit den zugehörigen Strukturebenen und den verschiedenen Phasen der Datenauswertung.




Der Übergang vom realen Auswertungsproblem, wie es sich für den Nutzer stellt, zur Auswertungsaufgabe, die die EDV-mäßigen Voraussetzungen berücksichtigt, entspricht einem Näherungsproblem. Die Existenz einer für den Nutzer akzeptablen Näherung wird im wesentlichen beeinflußt von

- der in der logischen Struktur zum Ausdruck kommenden Gestaltung des zur Auswertung vorgesehenen Datenbestandes (Inhalt, Umfang, Auswahl in Bezug zum Auswertungsproblem),
- der Struktur des Datenbestandes (Verfügbarkeit problembezogener Daten),

- der Leistungsfähigkeit des Auswertungssystems (Realisierung der Verknüpfungen und Darstellung).

Diese Näherung hat ihre Inversion beim späteren Übergang von der EDV-Lösung der Auswertungsaufgabe zur realen Problemlösung. Auf den ersten Blick hat dabei die Gestaltung der Daten als Ausdruck des Informationsgehaltes, der Informationsfülle primäre Bedeutung, denn je umfassender über bestimmte Prozesse informiert werden soll, um so umfassender müssen diese Prozesse durch Daten abgebildet werden. "Umfassend" bedeutet dabei aber nicht automatisch, daß eine Unzahl von Daten erfaßt und gespeichert werden, im Gegenteil. Mangelnde Überschaubarkeit wäre die Folge, und damit wären die Daten so gut wie begraben. Die Gestaltung der Daten wird damit zum Optimierungsproblem.

Ziel: Möglichst viel Information durch möglichst wenig Daten.

Die Verfügbarkeit, d.h. die Nutzbarmachung der Daten, ist aber wiederum nur über eine geeignete Struktur und entsprechenden Potenzen des Auswertungssystems möglich. Daten, die sich nicht umkehrbar eindeutig adressieren lassen, sind für die automatisierte Informationsverarbeitung im allgemeinen wertlos. (Klartext, mehrdeutige Datenkodierungen).

Andererseits aber gibt es unter den eindeutig identifizierbaren Daten die Teilmenge der fehlerhaften Daten. Die, falls sie nicht vom Auswertungssystem erkannt werden, die Lösung einer Auswertungsaufgabe beträchtlich verfälschen können. Die physische Struktur muß also die Potenz einer Wertung der Daten ermöglichen, denn das Weglassen der fehlerhaften Daten bedeutet einen Informationsverlust. Im ausgedruckten Zustand z.B. können bestimmte fehlerhafte Daten noch richtig interpretiert werden. Auch würde das bloße Weglassen dieser fehlerhaften Daten beim Aufbau des Datenbestandes nicht den Sachverhalt richtig widerspiegeln, denn es würde "ohne Angabe" bedeuten.

Zusammenfassend lassen sich aus dem Dargelegten einige konkrete Forderungen an die physische Struktur der zur Auswertung vorgesehenen Datenbestände ableiten:

1. Die physische Struktur aller vergleichbarer Datenbestände muß identisch sein. (Das heißt nicht, daß die jeweils konkrete Datenorganisation identisch ist. Auch die logische Struktur muß nicht identisch sein.)

2. Die Menge der Daten insgesamt und pro Bezugsebene (z.B. Patientenaufenthalt) muß in bestimmten Grenzen (Hardware-Problem) beliebig veränderbar sein.
3. Änderungen wie unter 2. dürfen nicht zur Veränderung der Identifizierungsadressen der nicht betroffenen Daten führen. (Sicherung der kontinuierlichen Qualität der Auswertung.)
4. Für alle einzelnen Elemente der Datenmenge müssen umkehrbar eindeutige Identifizeirungsadressen existieren.
5. Die Identifizierungsadressen sollten weitgehend aus dem Aufbau des Erfassungsmediums ableitbar sein.
   Logische Struktur = Datenorganisation
   (Hohe Überschaubarkeit, leichte Identifikation)
6. Feste Satzlängen sind ungeeignet.
   (Leerer Speicherplatz, hohe physische Laufzeit, keine Erweiterbarkeit über ein bestimmtes Maß hinaus)
7. Fehlerhafte Daten müssen kennzeichenbar sein.

Im Projekt "FRIEDA" des Krankenhauses Friedrichshain benutzen wir für die verschiedenen zur Auswertung vorgesehenen Datenbestände eine Struktur, die den wesentlichsten hier genannten Forderungen gerecht wird. Alle bei uns archivierten Datenmengen werden intensiv für die Lösung der verschiedensten Auswertungsprobleme genutzt.

Verfasser:
Dipl.-Math. Ing. Jürgen Melzer
Städtisches Krankenhaus im Friedrichshain
Organisations- und Rechenzentrum
1017 Berlin
Leninallee 49

Jantke, Renate

# Das Datenverwaltungssystem MAS/D-OS

## 1. Einleitung

Die gegenwärtige Entwicklung bei der Bewältigung von Aufgaben in verschiedenen EDV-Anwendungsbereichen fordert die Beherrschung einer Vielzahl informationstechnischer Einzelheiten. Daraus ergeben sich Forderungen an Datenverwaltungs- und Archivierungssysteme, die gleichartige Aufgaben, wie das Bereitstellen von Speicherplatz für Datenmengen, den Zugriff zu Daten, das Archivieren und Wiederauffinden, zu lösen haben.

Bei der Entwicklung und Implementierung des Datenverwaltungssystems MAS/D-OS (Memory Administration System/Directory-OS) als weiterentwickelte Version eines Systems für die BESM6 wurde das Ziel verfolgt, ein abgestimmtes System von Funktionen bereitzustellen deren einfache Anwendung es gestattet, Datenmengen effektiv und rationell zu verwalten und obengenannte Aufgaben zu realisieren.

Das Datenverwaltungssystem ist ein effektives Hilfsmittel in der Anwendungsprogrammierung, speziell auch für dialogorientierte Softwareentwicklung auf wissenschaftlich-technischem Gebiet geeignet.

Folgende Zielstellungen wurden bei der Entwicklung berücksichtigt:

- dynamische Bereitstellung von Speicherplatz für Datenmengen,
- effektiver Zugriff zu Daten, d. h. schneller Direktzugriff und sicherer Zugriff zu den Daten über Zugriffsfunktionen,
- nutzergesteuerte bzw. automatische Archivierung von Daten.

Dabei unterliegen die Datenmengen keinen Restriktionen, was ihre Struktur und ihren Inhalt betrifft.

Daraus ergibt sich die Möglichkeit, unterschiedlichste Informationen zu verwalten und zu archivieren, sowie dieses System als Basis für die Verarbeitung strukturierter Datenmengen zu nutzen.

Durch eine geeignete Implementation ist das Datenverwaltungssystem MAS/D-OS in unterschiedlichen Programmiersystemen nutzbar, in solchen mit Standardparametervermittlung OS/ES oder mit PASCAL 360 Parametervermittlung.

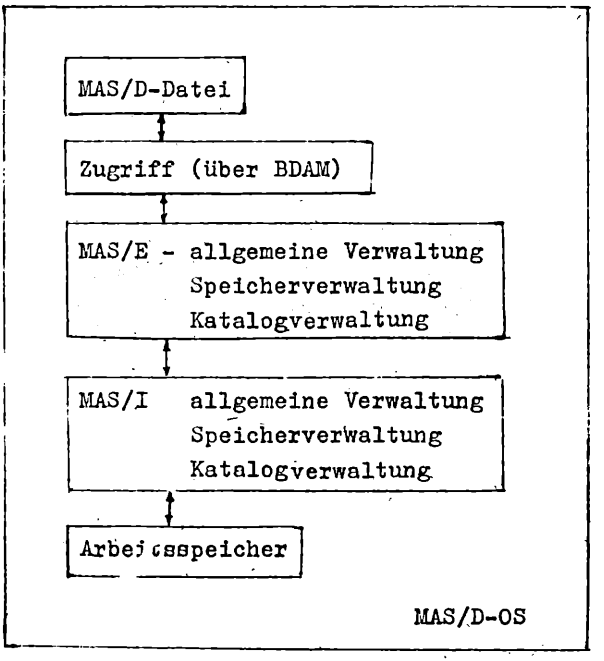
## 2. Systemarchitektur

Anhand der Systemarchitektur wird im folgenden die Arbeitsweise des Datenverwaltungssystems beschrieben.

Die zugrunde gelegten logischen Informationseinheiten sind Files, die als unstrukturierte Datenmengen beliebiger Länge aufgefaßt werden. Sie sind unter einem Namen - Filename - identifizierbar und durch einen Status - Filestatus - charakterisiert. Der Inhalt eines Files und sein Status sind voneinander unabhängig. Der Status steuert lediglich die Art und Weise der Arbeit des Systems mit diesem File.

Notwendige Voraussetzungen für die Arbeit des Systems sind eine Direktzugriffsdatei, die zur Aufnahme archivierter bzw. temporär ausgelagerter Files dient und ein Hauptspeicherbereich, der als Arbeitsspeicher für die dynamische Bereitstellung bzw. Freigabe von Speicherplatz für Files dient.

Auf diesen beiden Speicherbereichen sind Basisfunktionen, wie
Erzeugen der Datenstruktur
Operationen auf der Datenstruktur,
Löschen einer Datenstruktur
Statuseinstellung
und Archivierungsfunktionen für die in der Abbildung angegebenen Verwaltungsfiles erklärt, deren Gesamtheit für die interne Seite mit MAS/I und für die externe mit MAS/E bezeichnet wird.




Die Verwaltungen für MAS/I und MAS/E sind logisch voneinander getrennt und werden als Files aufgefaßt. Aus den Basisfunktionen, die auf diesen beiden Speicherbereichen erklärt sind, werden die bereitgestellten Nutzerfunktionen abgeleitet. Dadurch entsteht eine übersichtliche Programmarchitektur, die günstige Voraussetzungen für Änderungen bzw. Erweiterungen schafft.

Für folgende Prinzipien, die das Datenverwaltungssystem MAS/D-OS charakterisieren, stehen dem Nutzer Funktionen und Beschreibungsmittel zur Verfügung, auf die im nächsten Abschnitt hingwiesen wird.

- Für Files belieber Länge wird auf Anforderung Speicherplatz im Arbeitsspeicher bereitgestellt. Die Bereitstellung erfolgt entsprechend einem festgelegten Status (Ladestatus) in drei Formen:

(1) Zusammenhängender Speicherplatz wird bereitgestellt (sequential allocation).
(2) Speicherplatzbereiche, die über Referenzen verkettet sind, werden bereitgestellt (linked allocation).
(3) Es wird nur ein Speicherbereich fester Länge bereitgestellt, auf dem eine virtuelle Arbeitsweise durchgeführt wird (segmented allocation).

Die dynamische Verwaltung von Speicherplatz für Datenmengen, unterstützt insbesondere Anwendungen, bei denen der Bedarf an Speicherplatz erst während der Laufzeit von Programmen ermittelt wird.

- Files, die zu einem späteren Zeitpunkt wieder benötigt werden, können während der Laufzeit oder durch Abschluß der Arbeit in einer MAS/D-Datei auf einem externen Speicher archiviert werden. Nach erneuter Eröffnung der Arbeit mit dem System, stehen die archivierten Files zur weiteren Verarbeitung zur Verfügung.
- Der Zugriff zu Daten im File wird zweifach unterstützt. Zum einen ist es möglich zum Fileinhalt direkt zuzugreifen, zum anderen erfolgt der Zugriff über Zugriffsfunktionen. Durch den Direktzugriff wird ein schneller Zugriff realisiert. Allerdings fallen hier Kontrollen auf Zugriffsrechte, File- und Speichergrenzen weg, die beim Zugriff über Funktionen durchgeführt werden.

3. Nutzerschnittstellen

Sämtliche Funktionen des Systems MAS/D-OS stehen dem Nutzer als Unterprogramme zur Verfügung.
Die Strukturierung der MAS/D-Datei und die Initialisierung des Systems erfolgt durch eine allgemeine Funktion, die einmalig anzuwenden ist.

Den Rahmen für sämtliche Funktionen, die das Datenverwaltungssystem bietet, bilden eine Funktion zum Eröffnen der Arbeit und eine Funktion zum Abschließen der Arbeit. Innerhalb dieses Rahmens sind Zugriffs-, Auskunfts-, Bibliotheks- und andere Funktionen des Systems nutzbar.

Mit Hilfe von Filefunktionen kann man Files anlegen, ihren Status festlegen und Files nutzergesteuert laden bzw. entladen (archivieren). Die Zugriffsfunktionen lagern, falls es notwendig ist, Files prioritätengesteuert aus und laden benötigte Files automatisch in den Arbeitsspeicher. Über Auskunfts- und Bibliotheksfunktionen erhält man Angaben über die Charakteristika eines Files und der MAS/D-Datei. Bereitgestellte Änderungs- und Kopierfunktionen erweitern das Spektrum der Nutzerfunktionen. Mit dem Datenverwaltungssystem MAS/D-OS steht dem Anwendungsprogrammierer ein leistungsfähiges System als Hilfsmittel zur effektiven Softwareentwicklung zur Verfügung.

Literaturhinweise

/1/ Das Datenverwaltungssystem MAS/D-OS
Anwenderbeschreibung der PASCAL-360-Version
Arbeitsbericht am ZfR der AdW, Januar 1982

/2/ Das Datenverwaltungssystem MAS/D-OS
Anwenderbeschreibung der FORTRAN-Version
ZfR-Information, ZfR-NIE-82.00, März 1982

/3/ Das Datenverwaltungssystem MAS/D-OS
Forschungsbericht am ZfR der AdW, Juni 1982

Verfasser:
Dipl.-Math. Renate Jantke
Zentrum für Rechentechnik
der AdW der DDR
1199 Berlin
Rudower Chaussee 5

# Programme zur Leistungsbewertung im DVZ Dresden

ipl.-Ing. Elke Schreiber

Technische Universität Dresden

## 1. Einführung

In dem Maße, wie sich der Einsatz von Rechentechnik zur Lösung volkswirtschaftlicher Belange erhöht, gilt es den Datenverarbeitungsprozeß im Rechenzentrum noch stärker zu rationalisieren. Vor allem sind die für die Abarbeitung von Massendatenprojekten bereitgestellten Ressourcen effektiver zu nutzen. Zur Beseitigung der Diskrepanz zwischen den verfügbaren Ressourcen und ihrer Nutzung wurden zwei Programmsysteme implementiert mit dem Ziel der Bilanzierung und optimalen Abarbeitung von Projekten im DVZ Dresden:

1. Das Programmsystem SMFSTA zur Leistungsmessung von Arbeitslasten
2. Das Programmsystem MAPA zur Leistungsverbesserung durch Jobnetzoptimierung.

## 2. Verhaltensmodelle

Grundlage der Leistungsbewertung von Rechnerinstallationen bilden sogenannte Verhaltensmodelle. Sie erfassen den quantitativen Zusammenhang zwischen unabhängigen Einflußgrößen und Bewertungsgrößen und seiner Arbeitslast durch entsprechende Kenngrößen. Wesentliche Bewertungsgrößen sind:

- die Systemleistung L (Arbeitseinheiten/ZE)
- der Leistungsfaktor R

Zu den unabhängigen Einflußgrößen zählen:

- die Arbeitslast, wie sie durch die Vielzahl von Ressourcenanforderungen repräsentiert wird
- die Konfiguration, als Ausdruck verfügbarer Ressourcen je Geräteklasse
- sonstige Faktoren, z.B. die Ressourcenzuteilungsstrategie, wie sie in der Ablaufplanung genutzt wird.

Das Ausgangsmodell der Leistungsanalyse des Multiprogrammbetriebes beschreibt die Abhängigkeit des Leistungsfaktors R von bestimmten Einflußgrößen. R gilt als Maßzahl für die Bedienzeit eines Jobstapels über die Laufzeiteinheit des Rechners:

$$R = \frac{T_b}{T_h} \text{ , mit}$$

$T_b$ ,der Summenbedienungszeit aller Jobs und $T_h$ ,der Beschäftigungszeit des Jobstapels.

Es handelt sich dabei um ein Zweistufenmodell:

1. Stufe : Modell der festzugeordneten Ressourcen
2. Stufe : Modell der geteilt genutzten Ressourcen.

3. Modell der festzugeordneten Ressourcen

---

Das Modell der festzugeordneten Ressourcen kann durch folgende Formel beschrieben werden :

$$M = f(m,E(N),P) .$$

Danach ist der Multiprogrammfaktor M eine Funktion des Konfigurationsvektors m,des Arbeitslastvektors E(N) und der Reihenfolge der Jobbedienung P,wie sie im Ablaufplan festgelegt ist.Der Konfigurationsvektor

$$m = (m_S,m_P,m_B,m_D)$$

widerspiegelt die Zahl der festzugeordneten Ressourcen $m_i$ der Klasse i.Dabei gilt $i \in GK \{S,P,B,D\}$ ,wobei folgende Geräteklassen berücksichtigt werden:

S - Hauptspeicherplatz
P - Magnetplattengeräte
B - Magnetbandgeräte
D - Drucker.

Der Multiprogrammfaktor M ist für eine vorgegebene Gerätekonfiguration und ein vorgegebenes Aufgabenprofil nach oben begrenzt[1].Die ressourcenbedingte obere Grenze $M_g$ ergibt sich wie folgt:

$$M_g = \text{Min}\left\{\frac{m_i}{E(N_i)}\right\}$$

Über die Bestimmung von $M_g$ ergeben sich Möglichkeiten der Leistungserhöhung durch Anpassung der Arbeitslasten an vorhandene Konfigurationen.Die gewonnenen Ergebnisse sind Bestandteil der Kapazitätsplanung und nicht Gegenstand dieses Artikels.

Das Programmsystem SMFSTA dient der Erfassung und Auswertung von SMF-Daten und liefert Eingangsgrößen für das Programmsystem MAPA.Beide Programmsysteme leisten einen wesentlichen

Beitrag zur Leistungsverbesserung durch Jobnetzoptimierung von Datenverarbeitungsprojekten.

4. Meßwertauswertung von SMF-Daten

Grundlage der Auswertung bildet die Erfassung der Ressourcennutzung für drei Projekte der Tagesverarbeitung nach jedem Job- bzw.Jobschrittende in sogenannten SMF-Datensätzen auf einer Plattendatei. Von dieser Plattendatei werden täglich Bandabzüge erstellt, und die Sammlung von Bandabzügen über einen Monat wird der statistischen Auswertung zugrunde gelegt.
Ziel ist:

- die Ermittlung geeigneter Normativwerte als Eingangsgrößen für die Jobnetzoptimierung von Projekten
- die Bereitstellung von Meßwerten zur Bilanzierung der Projekte im DVZ Dresden.

Die Realisierung übernimmt das PS SMFSTA,das parametergesteuert die Auswertung von SMF-Daten vornimmt und die statistischen Größen in Form von Erwartungswerten und Verteilungsfunktionen beschreibt.Den problemorientierten Programmablaufplan des Programmsystems zeigt Abb.1.
Statistische Größen,die aufbereitet und verdichtet werden, sind u.a.:

- Jobverweilzeiten DJVZ
- Belastungsgrößen je Job HSM (Hauptspeicher-K-Byte-Minuten)
- Ressourcenbedarf der Jobs GERM (Geräteminuten pro Gerätetyp)

Eine Auswahl der pro Job ermittelten Normativwerte über statistische Größen für bestimmte Gerätetypen zeigt Tabelle 1
Die gewonnenen Meßwerte bilden die Grundlage für die Bestimmung der Teillasten $E(N_i)$ bei der Kapazitätsplanung.
Darüberhinaus liefert das PS SMFSTA Normativwerte der Jobverweilzeit,der Belastungsgrößen und des Ressourcenbedarfs der Jobs für das PS MAPA,einem Programm zur Jobnetzoptimierung von Datenverarbeitungsprojekten.

5. Leistungsverbesserung durch Jobnetzoptimierung

Ziel der Jobnetzoptimierung ist die Maximierung des Leistungsfaktors R bei der Abarbeitung von Projekten im Stapelbetrieb. Über eine optimale Gestaltung der Jobnetze,die den Projekten zugrunde liegen,wird eine Erhöhung des Multiprogrammfaktors möglich.
Ausgangspunkt der Untersuchungen ist die Erstellung eines optimalen Ablaufplans der Projektabarbeitung unter Beachtung d.

der Jobabhängigkeiten und der festzugeordneten Ressourcen. Zur Lösung dieser Aufgaben werden Algorithmen der Netzplantechnik mit beschränkten Ressourcen und Algorithmen der Strukturtheorie (erweitertes Matrizenkalkül) zur Bestimmung der Jobabhängigkeiten im PS MAPA genutzt. In Abb.2 ist die Grobstruktur des Programmsystems beschrieben.
Über die Ermittlung der Ein-und Ausgabedatein der Jobs aus den Job-und Prozedurbibliotheken lassen sich Vorgänger-und Nachfolgerelation der Jobs des Jobnetzes bestimmen. Diese Ergebnisse werden in Matrizenform abgespeichert. Zur Beschreibung der Struktur solcher Netze werden die aus der Netzplantechnik bekannten Vorgangsknotennetze verwandt. Für ein einfaches Jobnetz ist die Abhängigkeitsrelation in Abb.3 dargestellt.
Die Ausgabe der Jobabhängigkeiten durch das Programmsystem geschieht in verbaler Form.

```
Ausgabe der Jobabhängigkeiten:
  JOB A hat als Nachfolger JOB B
                           JOB J
  JOB B hat als Nachfolger JOB C
  JOB C hat als Nachfolger JOB D
                           JOB E

  JOB N hat keinen Nachfolger
```

Treten innerhalb des Netzes transitive Überbrückungen auf, so können diese durch Matrizentransponierung mit nachfolgender Abwärtsiteration beseitigt werden. Theoretische Grundlagen liefert dafür die angewandte Strukturtheorie[2].
Voraussetzung für die prioritätsgesteuerte Abarbeitung der Jobs ist eine Rangbestimmung. In Anlehnung an einen Algorithmus der Netzplantechnik[3] erfolgt nach einem bestimmten Modus eine Manipulierung der Zeilen-und Spaltenelemente der Vorgänger-Nachfolgematrix, in deren Ergebnis jeder Job einem Rang zugeordnet wird. Parallel abzuarbeitende Jobs gehören einem Rang an ( Abb.3).
Unter Berücksichtigung festzugeordneter Systemressourcen

erfolgt eine Eingliederung sämtlicher Jobs nach entsprechenden Prioritätskriterien in einen terminierten Ablaufplan. Die Optimierung erfolgt rangweise, so daß Jobs des 1. Ranges höchste Priorität aufweisen. Innerhalb der Jobs eines Ranges erfolgt eine weitere Prioritätsauswahl:

- Jobs, die im nächsten Rang einen Nachfolger haben, erhalten Abarbeitungsvorrang vor denen ohne Nachfolger
- Jobs mit längster Restverweilzeit erhalten höchste Priorität .

Es erfolgt eine weitere Prioritätssetzung bei der Eingliederung der Jobs in den terminierten Ablaufplan. Dabei wird geprüft, ob eine Einordnung in die hauptspeichergünstigste Partition möglich ist, sonst wird der Job in die pufferzeitgünstigste Partition eingeordnet.

Im Ergebnis druckt das PS MAPA einen Partitionbelegungsplan, aus dem der Operator die Reihenfolge der Jobabarbeitung, die dafür aufzuwendende Zeit [min] und die Belegung der Verarbeitungspartition erkennen kann.

Das PS MAPA wurde an spezifischen Projekten im DVZ Dresden erprobt und brachte Laufzeiteinsparungen von 6 ~ 15%. Ein Vergleich der Multiprogrammfaktoren M und Leistungsfaktoren R in Tabelle 3 zeigt für drei konkrete Projekte eine erhebliche Leistungsverbesserung bei den optimierten Varianten.

Literaturverzeichnis:


1] Bergholz, G.: Zur Leistungsanalyse von Rechnersystemen der Stapelverarbeitung
Schriftenreihe des WBZ für MKR/IV, Heft 43/80

2] Stoschek, E. Angewandte Strukturtheorie
Akademie-Verlag, Berlin 1981

3] Weber, H.D.; Wille, H.; Gewald, K. : Netzplantechnik- Methoden zur Planung und Überwachung von Projekten
R. Oldenbourg-Verlag, München 1972

Tabelle 1 : Normativwerte über statistische Größen für bestimmte Gerätetypen

| Auftrags-Nr. | Job-name | DCPU | DJVZ | HSM | 5o1o | 5o61 | LKL | PD | N |
|---|---|---|---|---|---|---|---|---|---|
| AN 1 | JOB1 | o,19 | 38,63 | 738,8 | 40,25 | 1,00 | 39,63 | 0,00 | 26 |
| AN 2 | JOB2 | 0,06 | 3,56 | 202,9 | 0,25 | 15,69 | 0,00 | 0,00 | 26 |
| AN 3 | JOB3 | 0,00 | 0,25 | 5,31 | 0,00 | 0,06 | 0,00 | 0,00 | 23 |
| AN 4 | JOB4 | 0,06 | 3,44 | 110,6 | 11,56 | 0,00 | 23,25 | 0,00 | 16 |
| AN 5 | JOB5 | 15,25 | 27,06 | 184,8 | 77,88 | 153,6 | 0,00 | 0,00 | 22 |
| AN 6 | JOB6 | 0,38 | 4,38 | 323,9 | 0,00 | 28,19 | 0,00 | 0,00 | 23 |
| AN 7 | JOB7 | 0,13 | 3,38 | 53,81 | 4,25 | 5,44 | 0,00 | 0,00 | 22 |
| AN 8 | JOB8 | 0,13 | 5,06 | 160,2 | 4,13 | 0,56 | 0,00 | 4,13 | 26 |

Tabelle 2 : Leistungsverbesserung konkreter Projekte

| Projekte | M | | R | | $\frac{R_{alt}}{R_{neu}}$ |
|---|---|---|---|---|---|
| | alt | neu | alt | neu | |
| KORW | 1,00 | 1,17 | 1,00 | 1,12 | 1,12 |
| GETRÄNKE | 1,34 | 1,58 | 1,21 | 1,35 | 1,12 |
| FISCH | 1,45 | 2,03 | 1,24 | 1,47 | 1,18 |

Tabelle 3 Partitionbelegungsplan

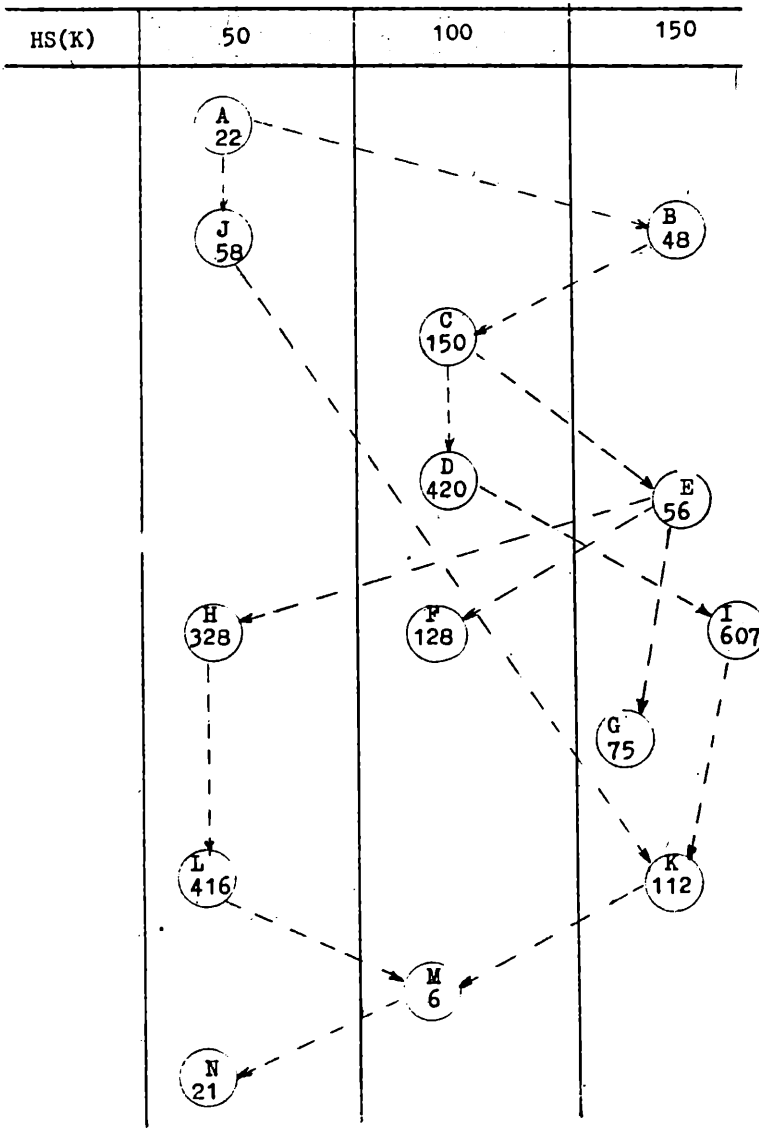


Abb. 1 Problemorientierter Programmablaufplan des PS SMFSTA

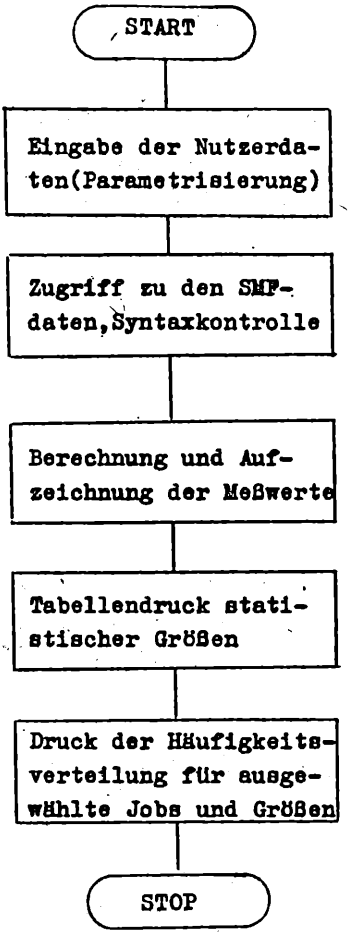


Abb. 2 Grobstruktur des PS MAPA

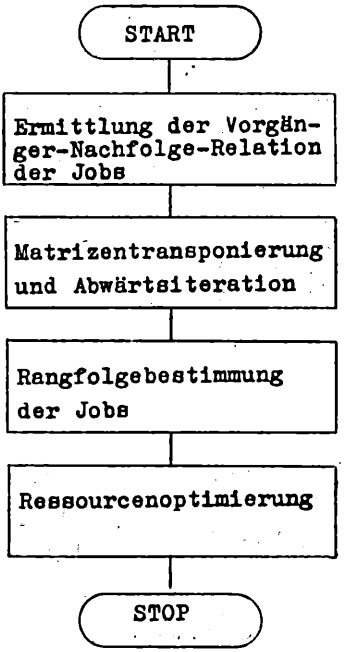
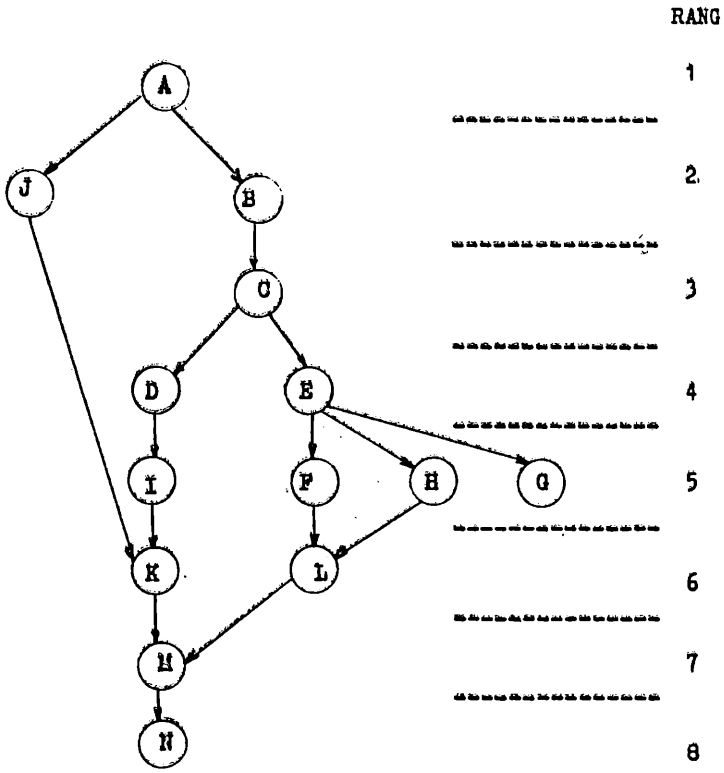


Abb. 3 Abhängigkeitsrelation eines Jobnetzes



Sadowski, Horst

## Verfahren zur Automatisierung der Verarbeitungstechnologie

### 1. Ansprüche an eine moderne EDV-Technologie

Bei der Entwicklung von Verfahren zur Automatisierung der Verarbeitungstechnologie wurde von folgenden Ansprüchen ausgegangen, die heute an eine moderne EDV-Technologie gestellt werden:

- den Anwender weitestgehend von Problemen der Verarbeitungstechnologie zu befreien
- im Rechenzentrum den Produktionstechnologen und Maschinenbedienern betriebssichere, einfache und einheitliche technologische Hilfsmittel in die Hand zu geben
- die Auftragsbearbeitung im Rechenzentrum zu rationalisieren und die vorhandene EDVA mit hoher Effektivität zu nutzen
- die Programmtestung und Erprobung unter Produktionsbedingungen durchführen zu können
- Möglichkeiten für einen vom Anwender gesteuerten Dialogbetrieb vorzusehen.

### 2. Das allgemeine Steuerprinzip der Stapelverarbeitung

Eine Methode der Verarbeitungstechnologie ist die Stapelverarbeitung. Das Prinzip der Stapelverarbeitung besteht darin, daß in einem sogenannten Anwendungsfall die Nutzung eines Projektes durch die Anwendung bestimmter problemorientierter Systemunterlagen (im folgenden kurz POS genannt) auf anwendungsfalleigene Datenbestände erfolgt.

Jede Projektnutzung für einen Anwendungsfall wird als Arbeitsauftrag formuliert und besteht neben manuellen Arbeiten zur Datenbereitstellung und ähnlichem vor allem aus Operationen, die auf der EDVA unter der Steuerung eines Steuerprogramms im Rahmen eines Jobs in einem oder mehreren Jobschritten zur Ausführung gelangen. Solche Operationen sind z.B. Pflege, Rechnung, Listung. Die Operationen werden vom Anwender vorgeschrieben und durch die Angabe von Parametern in ihren Funktionen gesteuert.

Zur Einsparung von Anlagenkapazität, zur Vereinheitlichung der Parameterübergabe für die Operationen, zur Vereinfachung der Programmierung und der Technologie der Nutzung von POS wurde ein allgemeines Steuerprinzip erarbeitet.

Die Ausführung einer Operation einer POS wird durch eine sogenannte Operationsanweisung veranlaßt. Die Operationsanweisung hat den folgenden prinzipiellen Aufbau:

./[name] operation operanden

name - wahlfrei, kann zur Kennzeichnung der Operationsanweisung im Protokoll benutzt werden

operation - Angabe der Operation aus einer POS
z.B.: Datenpflege, Rechnung, Listung

operanden - Schlüsselwort- oder Stellungsparameter, dienen zur Steuerung der Operation und zur Definition der zu verwendenden Datenbestände
z.B.: Geheimhaltungsgradangaben für Drucklisten,
zu verwendende Pakete von Rechenformeln,
Auswahlbedingungen,
Datenbestandsnamen

Für die wiederholte Abarbeitung einer Folge von Operationsanweisungen kann diese Folge zur Vereinfachung für den Anwender zur Makrooperation zusammengefaßt werden. Makrooperationen können vom Anwender selbst definiert werden und haben den folgenden Aufbau:

.%[name] moper operanden

name - s. o.

moper - Name der Makrooperation

operanden - Parameter, die durch die Makrooperation an die in ihr enthaltenen Elementaroperationen weitergereicht werden

Funktionen des allgemeinen Steuerprinzips

- Operationsanweisung bewirkt die Abarbeitung eines Programmes
- Verarbeitung von Makrooperationen möglich, indem diese in Elementaroperationen überführt werden
- auf Operationsanweisungen können unmittelbar Subanweisungen zur Steuerung der Operation oder Daten folgen, sofern die entsprechende Operation das vorsieht
- bedingte Abarbeitung in einer Folge von Operationsanweisungen durch die Auswertung des erfolgreichen Abschlusses der vorhergehenden Operationsanweisungen oder des Zustandes der zu verwendeten Datenbestände möglich

- Wiederanlauf im Falle eines Abbruchs der Verarbeitung eines Auftrages möglich
- syntaktische Kontrolle der Operationsanweisungen ohne Abarbeitung der Programme möglich

Für die Realisierung des beschriebenen Steuerprinzips sind die folgenden ausgezeichneten Anweisungen notwendig:

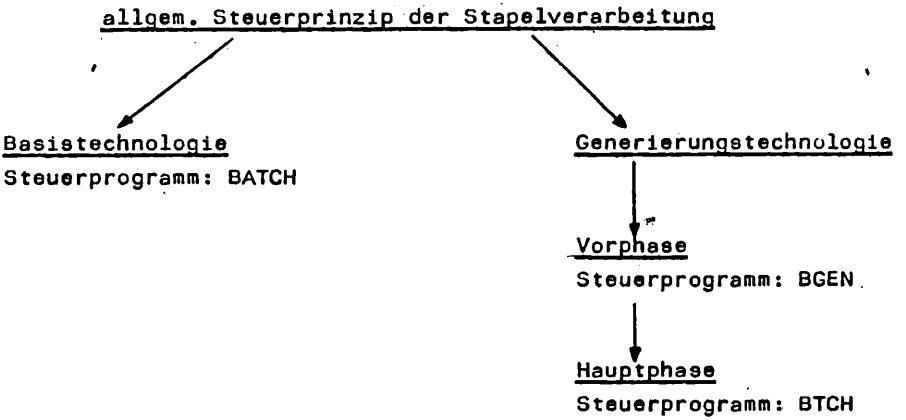
```
.+ NAME  name
         zur Vorbereitung der Wiederanlaufmöglichkeit
.+ RESTART  name
         zur Ausführung des Wiederanlaufs
.+ PARM  parameter
         zur Parameterübergabe an das Steuerprogramm
         z.B. Geheimhaltungsgrad des Ablaufprotokolls,
              Bedingungen für den Abbruch der Verarbeitung
```

Für die Nutzung der Wiederanlaufmöglichkeit ist die Bereitstellung eines Referenzspeichers Voraussetzung.

Beispiel für eine Folge von Steuer- und Operationsanweisungen:

```
.+ NAME  RSANW                  werden vom Technologen im RZ
.+ PARM  C=12,GGN=NFD           hinzugefügt
-------------------------------------------------------------
./ ADD   EIN=*                                Operationsanwei-
   : Kartenstapel, der von ADD verar-         sungen eines Auf-
   : beitet werden soll                       trages, stammen
                                              vom Anwender
./ PFL   EIN=STAMM,AUS=LSTØ1,VAR=(A,B)
./ RECH  EIN=LSTØ1,AUS=AUS,FPAKA=(FPAK5999,GAUSW=(5))
./ LIST  EIN=AUS
```

Das allgemeine Steuerprinzip der Stapelverarbeitung ist nun die Grundlage für zwei Regime der Verarbeitungstechnologie.




## 3. Basistechnologie

Dieses Regime ist dadurch ausgezeichnet, daß der Anwender für einen Arbeitsauftrag einen komplett zusammengestellten Job bereitstellt, der unmittelbar auf der EDVA abgearbeitet werden kann. Ein solcher Job hat z.B. das folgende Aussehen:

```
//RECHØ1   JOB  ...
//STEPØ1   EXEC  PGM=BATCH
//STEPLIB  DD   ...
//BASPRINT DD   ...
//BAREFS   DD   ...
//BALIST   DD   ...
//AUS      DD   ...
//SORTOUT  DD   ...
//BASIN    DD   *
./ RECH  EIN=SORTOUT,AUS=AUS,FPAKA=(FPAK5999,GAUSW=(5))
```

Vom Technologen werden dann nur noch die PARM- und NAME-Anweisungen hinzugefügt, falls es erforderlich ist.
Dieses Technologieregime hat aber folgende Nachteile:

- Anwender muß sämtliche Jobsteueranweisungen liefern
- Problem der richtigen Zuordnung von Datenbeständen zu Datenträgern bei einer Vielzahl von Datenbeständen und mehreren Generationen
- Bezugnahme auf Datenbestände nur über die Angabe der bereitgestellten Jobsteueranweisungen

. Bedingte Ausführung von Operationsanweisungen nur in Abhängigkeit von der erfolgreichen Bearbeitung vorhergehender Anweisungen; Auswertung des Datenbestandszustandes nicht möglich

Daraus ergibt sich, daß die Basistechnologie nur für die Bearbeitung von aus technologischer Sicht anspruchslosen Projekten mit wenigen, kaum wechselnden Datenträgern und für vom Programmierer betreute Testrechnungen eingesetzt werden sollte.

## 4. Generierungstechnologie

Um die Nachteile der Basistechnologie zu beseitigen, wurde die Generierungstechnologie entwickelt. In diesem Regime sind vom Anwender nur die Operationsanweisungen bereitzustellen. Die Generierungstechnologie unterteilt sich in zwei Phasen.

### 4.1. Vorphase BGEN

Mittels BGEN wird der gesamte vom Anwender bereitgestellte Eingabestrom den folgenden Bearbeitungsschritten unterzogen:

Auflösung von Makrooperationen in Elementaroperationen, sofern welche vorhanden sind

- Syntaktische Analyse der Operationsanweisungen
- Überprüfung auf Zulässigkeit und Vollständigkeit der Parameter zu technologischen Angaben
- Überprüfung auf Zulässigkeit und Existenz benötigter Speicher
- Erzeugung benötigter Jobsteueranweisungen in einem oder mehreren Jobschritten und Zuordnung der Operationsanweisungen zu den Jobschritten
- Formatisierung eines Registrierobjektes im Referenzspeicher
- Erzeugung einer Jobbeschreibung für den in der Phase BTCH zu startenden Verarbeitungsjob

Bei BGEN erfolgen noch keine Manipulationen mit den Datenbeständen.

### 4.2. Hauptphase BTCH

In dieser Phase erfolgt unter dem Steuerprogramm BTCH die Abarbeitung der Operationsanweisungen mit dem in der Vorphase BGEN erzeugten Job.

Entsprechend den in den Operationsanweisungen enthaltenen technologischen Angaben weist BTCH den Verarbeitungsprogrammen die

zu verwendenden DD-Namen für die Jobsteueranweisung, durch die die zu verarbeitenden Datenbestände definiert werden, zu.
Unter dem Regime der Generierungstechnologie wird für jeden Datenbestand eine sogenannte Datenbestandscharakteristik automatisch geführt. In ihr werden unter anderem vermerkt:

Name und Nummer der Operationsanweisung, durch die der Bestand erstellt oder gepflegt wurde
inhaltlicher und technischer Zustand des Bestandes
- Name der Dateibeschreibungstafel für den Bestand
- Sortierung des Bestandes
- Anzahl der Datensätze

BTCH unterstützt die Verarbeitungsprogramme weitestgehend bei der Auswertung und Pflege der Datenbestandscharakteristik und übernimmt gegebenenfalls automatisch eine benötigte Vor- oder Nachsortierung der Datenbestände.
BTCH registriert laufend den Bearbeitungszustand und ermöglicht dadurch einen automatischen oder gesteuerten Wiederanlauf der Abarbeitung der Operationsanweisungen.
Die Vorteile der Generierungstechnologie gegenüber der Basistechnologie bestehen nun im folgenden:

- Anwender muß nur die Operationsanweisungen liefern
- die Zuordnung von Datenbeständen zu Datenträgern und die Erzeugung der Jobsteueranweisungen erfolgt automatisch
- in den Operationsanweisungen kann nur durch die Angabe des Datenbestandsnamens bzw. des Speichernamens Bezug auf die Datenbestände genommen werden
- durch die Datenbestandscharakteristik
  - Steuerung der bedingten Ausführung von Operationsanweisungen durch die Auswertung des Zustandes der Datenbestände
  - Reduzierung der Rechenzeit durch Vermeidung unnötiger Datenbestandssortierungen
  - bessere Überwachung des Bearbeitungszustandes von Arbeitsaufträgen

Voraussetzung für die Arbeit im Regime der Generierungstechnologie sind:

die Bereitstellung eines Referenzspeichers

die Definition von Speichern mittels des Systems DAMODASS
die Bereitstellung eines sogenannten Anwendungsfallmoduls zur Konkretisierung allgemeiner technologischer Anforderungen

Verfasser:
Dr. Horst Sadowski, Berlin

Dipl.-Math. Günther Kroß, Humboldt-Universität

# SPOOL 2.0 - System zur Unterstützung eines zeitweise bedienungsunabhängigen Betriebes an ESER-Rechnern

## 1. Vorbemerkungen

Das System SPOOL 2.0 ist eine ab Januar 1982 verfügbare Version des Systems SPOOL, dessen Ziel die Rationalisierung und Automatisierung der Produktion an Rechenzentren mit ESER-Anlagen ist (vgl. /1/). SPOOL 2.0 ist als Betriebssystemkomponente unter den Steuerprogrammkonfigurationen MVT und MFT des OS/ES einsetzbar. Die spezielle Zielstellung für dieses System ist die Ermöglichung eines bedienungsarmen bzw. eines zeitweise bedienungsunabhängigen Betriebes der EDVA und somit insbesondere eine Steigerung der Arbeitsproduktivität hinsichtlich des Verhältnisses von Bedienungsaufwand (Anzahl der Bedienkräfte) zu Produktionsvolumen (produktive Laufzeit). Das System geht von dem Grundgedanken aus, die Erreichung dieses Zieles durch eine Erleichterung der Bedienarbeit, durch eine Konzentration der Bedienarbeit auf bestimmte Zeitintervalle und damit verbunden durch die Schaffung von Intervallen bedienungsunabhängiger Produktion zu unterstützen. Dabei wird davon ausgegangen, daß bedienungsunabhängige Zeiten, also die Zeiten in denen die Produktion an der EDVA ohne Bedienereingriff gesichert ist, ohne zusätzliche Belastung anderer Bereiche des Rechenzentrums, wie z.B. die Arbeitsvorbereitung und bei einem normalen Stapeljobprofil, also z.B. nicht notwendig mit länger laufenden Jobs, für täglich mehrere Stunden zu gewährleisten sind.

Gegenüber seinen Vorgängerversionen besitzt das System SPOOL 2.0 einige neue, seine speziellen zusätzlichen Funktionen realisierende Komponenten. Das System ist in der Lage, in verschiedenen Betriebsweisen, die sich vor allem hinsichtlich der Jobauswahl, der Konsolenunterstützung und der Bedieneraktivität unterscheiden, zu arbeiten. Voraussetzung für die Gewährleistung der unterschiedlichen Betriebsweisen ist dabei eine Jobklassifizierung hinsichtlich deren Bedienanforderungen.

## 2. Funktionsprinzipien des Systems SPOOL 2.0

### 2.1. Klassifizierung der Jobs

SPOOL 2.0 unterteilt die eingelesenen Jobs in die drei im folgenden aufgeführten, sich hinsichtlich der Bedienanforderungen unterscheidenden Kategorien:

a) bedienungsunabhängige Jobs:
   - Keine Magnetbandarbeit,
   - Keine Benutzung von mehr als zwei privaten Magnetplatten,
   - Kein Direktanspruch peripherer Geräte,
   - Keine Modifikation verfallsgeschützter Dateien,
   - Keine Bedienerkommunikation;

b) bedingt bedienungsunabhängige Jobs:
   wie a), wobei zusätzlich erlaubt sind
   - Arbeit mit privaten Magnetbändern,
   - Arbeit mit mehr als zwei privaten Magnetplatten;

c) bedienungsabhängige Jobs:
   Komplementärmenge zu a) und b).

Die Jobklassifizierung erfolgt mittels der Analyse der durch die Nutzer in ihren Jobs bereitzustellenden SPOOL-Steueranweisungen /*JOBPARM und /*SETUP. In ihnen müssen die benötigten Magnetdatenträger, die direkt angesprochenen Geräte, die zu modifizierenden verfallsgeschützten Dateien und ggf. die Verwendung von WTOR's im Job angegeben werden.

### 2.2. Betriebsweisen des Systems SPOOL 2.0

Über ein Kommando der SPOOL-Kommandosprache ansteuerbar ist das System in der Lage, in drei Betriebsweisen zu arbeiten:

B1: Abarbeitung von bedienungsabhängigen und bedingt bedienungsunabhängigen Jobs;

B2: Abarbeitung von bedienungsunabhängigen Jobs sowie derjenigen bedingt bedienungsunabhängigen Jobs, deren Magnetdatenträger montiert sind (MOUNT für private Magnetbänder) und die mittels eines Kommandos als bedienungsunabhängig gekennzeichnet wurden;

B3: Abarbeitung der Jobs unabhängig von deren Klassifizierung.

Allen Betriebsweisen gemeinsam ist eine magnetplattenabhängige Jobauswahl. Es können generell nur solche Jobs zur Verarbeitung ausgewählt werden, die entweder keine privaten Magnetplatten verwenden oder deren Magnetplatten dem System über ein spezielles Kommando als logisch verfügbar mitgeteilt wurden und die in der praktischen Verfahrensweise i.a. vorher auch physisch montiert wurden. Diese magnetplattenabhängige Jobauswahl erleichtert die Bedienung und beugt Bedienfehlern und den daraus gegebenenfalls resultierenden Wartezuständen bei Montageanforderungen vor. Während der Betriebsweisen B1 und B3 ist es möglich, die Datenträgerabhängigkeit der Jobauswahl per Kommando zu entaktivieren.

Standardbetriebsweise ist die bedienungsabhängige Betriebsweise B1. In ihr werden nur Jobs abgearbeitet, von denen angenommen wird, daß sie Bedienanforderungen hervorrufen. Bedienungsunabhängige Jobs werden nicht ausgewählt. Diese können somit während der Betriebsweise B1 für einen bevorstehenden bedienungsunabhängigen Betrieb zurückgehalten werden.

Eine bedienungsunabhängige Arbeit der EDVA ist durch Anwendung der bedienungsunabhängigen Betriebsweise B2 möglich. Während dieser werden vom System SPOOL 2.0 nur die Jobs zur Verarbeitung ausgewählt, deren Magnetplatten verfügbar sind und die keine Bedienanforderungen hervorrufen. Das sind bedienungsunabhängige Jobs und diejenigen bedingt bedienungsunabhängigen Jobs, für die der Bediener vor der Aktivierung der Betriebsweise B2 eine Bedienungsunabhängigkeit sichern kann, wie z.B. durch MOUNT-Kommandos für die benötigten privaten Magnetbänder. Während der Betriebsweise B2 sind sämtliche langsamen und bedienungsintensiven E/A-Geräte wie Kartenleser, Drucker u.a. inaktiv. Die Druckausgabe der Jobs wird mit entsprechenden Listenkennzeichen zur späteren Separierung während der Betriebsweisen B1 oder B3 versehen auf einem Magnetband zwischengespeichert. Die Kommunikation Betriebssystem - Konsole wird durch ein spezielles Konsolinterface übernommen. Dieses Interface analysiert sämtliche Konsolnachrichten und übernimmt eine gegebenenfalls notwendige Beantwortung. Dabei werden Jobs, die Bedienanforderungen hervorrufen durch CANCEL aus dem System entfernt. Betroffen werden können dadurch i.a. nur solche Jobs, die unvollständige Angaben in den SPOOL-Steueranweisungen haben. Die Arbeit der Kon-

sole selbst wird dadurch weitgehend entaktiviert, daß bis auf spezielle Systemnachrichten und Antworten auf Bedienerdisplaykommandos sämtliche Nachrichten lediglich in das Systemprotokoll ausgegeben werden.

## 2.3. Übergang zur bedienungsunabhängigen Betriebsweise

Der Übergang von einer bedienungsabhängigen Betriebsweise (B1, B3) zur bedienungsunabhängigen Betriebsweise B2 wird durch ein Kommando ($TS,FREE) initiiert und durch einen wechselseitigen Kommunikations- und Handlungsprozeß SPOOL-Bediener realisiert. Alle notwendigen Maßnahmen werden vom System angefordert und überprüft, um eine darauffolgende reibungslose und effektive bedienungsunabhängige Arbeit zu garantieren. Bei der folgenden Skizzierung des Umschaltprozesses bezeichne $B_i$ eine Bedienerhandlung und $S_i$ die Handlungsfolge von SPOOL 2.0.

$B_1$: Kommando $T...,FREE

$S_1$: - Stoppen der Jobauswahl und der SPOOL-E/A-Geräte (LKL, PD usw.);
- Warten auf die Beendigung der Abarbeitungsphase ggf. noch aktiver bedienungsabhängiger Jobs;
- Initiierung einer Bedienerinformationsliste (Kommando $L).

$B_2$: Steuerung der Ausgabe der Bedienerinformationsliste, die jobbezogenen Angaben wie Datenträger, Geräte, zu überschreibende verfallsgeschützte Dateien u.a., sowie magnetplattenbezogenen Angaben wie Anzahl der sich auf eine Magnetplatte beziehenden Jobs und Gesamtlaufzeit dieser Jobs (unterteilt nach Bedienungsabhängigkeit bzw. Bedienungsunabhängigkeit) enthält und somit geeignet ist, um insbesondere eine günstige Datenträgerkonfiguration zu bestimmen.

$S_2$: "OUTPUT TAPE REQUIRED ..."

$B_3$: Start einer Magnetbanddruckausgabe: $STPEi,adr

$S_3$: "USER DISKS MUST BE RESERVED ..."

$B_4$: - Überprüfung und ggf. Veränderung der Magnetplattenkonfiguration mittels der Informationen aus der Bedienerinformationsliste und Bedienerdisplaykommandos. Die für die bedienungsunabhängige Betriebsweise vorgesehenen Magnetplatten müssen den Status „reserviert" haben.
- Fertigmeldung durch $TS,FREE.

$S_4$: - Mitteilung über verfügbare Magnetplatten;
- "nn SYSTEM READY ... REPLY Y OR N"

$P_5$: - Antwort N veranlaßt eine Unterbrechung des Umschaltprozesses, in der der Bediener ggf. noch Änderungen der Datenträgerkonfiguration vornehmen kann, insbesondere private Magnetbänder durch MOUNT-Kommando montieren kann und bedingt bedienungsunabhängige Jobs in den Zustand bedienungsunabhängig versetzen kann.
- Antwort Y veranlaßt die Beendigung des Umschaltprozesses durch das System.

$S_5$: Aktivierung des Konsolinterface und der Jobauswahl.

3. Bemerkungen zum Nutzen des Systems

Der durch den Einsatz des Systems SPOOL 2.0 zu erreichende Nutzen liegt auf mehreren Ebenen und ist von verschiedenen Faktoren abhängig. Allgemeine Betrachtungen zum Nutzen des Systems SPOOL sind /1/ zu entnehmen. Hier herausgegriffen seien nur die Effekte, die durch die oben dargestellten Komponenten des Systems erreicht werden.

Generell kann der Effektivitätsgewinn und die Bedienerfreundlichkeit durch den allgemeinen Einsatz der magnetplattenabhängigen Jobauswahl genannt werden. Dem Bediener ist es mittels der ihm durch das System gegebenen umfangreichen und detaillierten Informationsmöglichkeiten leicht möglich, effektive Datenträgerkonfigurationen zu geeigneten Zeitpunkten zu bestimmen; Wartezeiten durch nicht vorhergesehene bzw. Ausfallzeiten durch nicht erfüllbare Montieranforderungen (MP) entfallen.

Ein weiterer ebenfalls insbesondere den Anlagendurchsatz positiv beeinflussender Faktor ist die besonders effektive, da von jeder Kommunikation und Bedienerhandlung sowie von der Aktivität langsamer E/A-Geräte befreite Arbeit der EDVA während der bedienungsunabhängigen Betriebsweise.

Die Erhöhung der Arbeitsproduktivität durch eine Konzentration der Bedienhandlungen auf bestimmte Zeiträume und das mögliche Betreiben der EDVA mit vermindertem Bedienpersonal im bedienungsunabhängigen Betrieb wurde schon einleitend erwähnt. Der hierdurch erzielte Nutzen wird vom Umfang des Einsatzes der bedienungsunabhängigen Be-

triebsweise bestimmt.

Die tägliche Dauer der bedienungsunabhängigen Zeitintervalle ist von verschiedenen Faktoren, wie Jobprofil, Gerätekonfiguration (MP, MB), Verteilung der Magnetplattendateien u.a. abhängig. Sie beträgt am Organisations- und Rechenzentrum der Humboldt-Universität durchschnittlich 3-4 Stunden, wobei Spitzenwerte insbesondere an den Wochenenden von 7-10 Stunden erreicht werden. Diese Werte sind hier erweiterungsfähig, da der Anteil der bedienungsunabhängigen Jobs durchschnittlich über 60 % beträgt. Bezüglich der Joblaufzeiten ist der Anteil der bedienungsunabhängigen Jobs noch größer. Die bedienungsunabhängige Betriebsweise wird dabei durch einen entsprechenden Schichtplan unterstützt und wird verwendet, um die EDVA zu bestimmten Produktionszeiten mit stark verringertem Bedienpersonal zu betreiben. Dem Bedienpersonal kommt dabei lediglich die Aufsichtspflicht und der Eingriff bei Havarien zu. Betont werden soll an dieser Stelle, daß das System SPOOL 2.0 in seiner bedienungsunabhängigen Betriebsweise systemseitig die Produktion auch ohne Bedieneranwesenheit sichern kann.

Literaturhinweise

/1/ Winks, Hans
"SPOOL - ein System zur Automatisierung von Produktionsprozessen an ESER-Rechnern"

Meißler, Hans-Günter

# SPOOLRJE - eine Fernverarbeitungskomponente des SPOOL-Systems

## 1. Einleitung

Aufbauend auf die Prinzipien des Arbeitsvorbereitungssystems SPOOL wird mit SPOOLRJE eine Möglichkeit der Jobfernverarbeitung an ESER-Anlagen realisiert.

Folgende Aspekte sollen dabei berücksichtigt werden:

- Die Vorteile der Stapelverarbeitung bleiben für den Betreiber der EDVA erhalten.
  Die Jobeingabe von entfernt aufgestellten Terminals an einen Zentralrechner, ihre Verarbeitung dort und die wahlweise Ausgabe der Ergebnisse der Jobs am Terminal oder am Zentralrechner gestatten weiterhin die Vergabe von Prioritäten und Ressourcen nach langjährigen und bewährten Verfahren, die eine bestmögliche Auslastung der EDVA sichern sollen.
  Das Rechenzeitregime, der Bediener oder entsprechende Überwachungsprogramme des Zentralrechners bestimmen den Zeitpunkt der Jobannahme, Abarbeitung und Ausgabe. Das dem Betriebssystem beigeordnete Arbeitsvorbereitungssystem SPOOL unterstützt diese, die ursprüngliche Arbeitsvorbereitung übergehende, Arbeitsweise der Jobeingabe durch die Angabe von vielen Jobparametern, die die Ressourcenbelastung des Jobs ausweisen.

- Dem Nutzer des Zentralrechners bieten sich erweiterte Möglichkeiten.
  Durch die Verwendung von Terminals, die am Arbeitsplatz des Nutzers oder in seiner unmittelbaren Nähe aufgestellt werden, verringern sich die Zeiten von der Formulierung des Problems über die Kodierung bis zur Eingabe in die EDVA beträchtlich.
  Es entfallen Wegezeiten, die Inanspruchnahme von Datenerfassungsleistungen und die noch übliche manuelle Kontrolle auf Verwendungsdauer und Art der im Job benötigten Ressourcen.
  Im Gegensatz zu Dialogsystemen, die in begrenztem Umfang auch eine Jobfernverarbeitung gestatten, muß der Nutzer keine neue

Sprache zur Jobeingabe erlernen. Es genügt die ihm bisher bekannte Jobsteuersprache des OS/ES. Viele bekannte Aufgabenstellungen des Nutzers erfordern keinen direkten Dialog mit dem Zentralrechner; sie fallen in die auch stapelmäßig abarbeitbare Kategorie der Abfragen.
Bestimmte Terminals gestatten dem Nutzer nach Absenden der Jobs kleinere Probleme offline, d. h. getrennt vom Zentralrechner zu lösen.
Wahlweise kann der Nutzer nur durch die Eingabe bestimmter Kommandos über TSO oder über SPOOLRJE mit dem Zentralrechner verbunden sein.

## 2. Terminals und Leitungen

SPOOLRJE unterstützt in verschiedenen Ausbaustufen unterschiedliche Terminaltypen.
Eine erste Ausbaustufe orientiert vorrangig auf reine Dialogterminals, z. B. Typen wie AP64 oder BS7920. Hierbei ist nicht der spezielle Terminaltyp, sondern seine Nichtprogrammierbarkeit entscheidend für die Einordnung in diese Gruppe. Als Leitungen sollen vorrangig Standleitungen mit asynchroner oder synchroner Übertragung und Übertragungsgeschwindigkeiten bis zu 1200 Bd verwendet werden. Ein Übergang zu Wählleitungen ist geplant und soll zuerst am Datenhandvermittlungsnetz und im Telexnetz der Deutschen Post funktionell erprobt werden. Dazu werden auch Fernschreiber an Wählleitungen in dieser Ausbaustufe in SPOOLRJE mit einbezogen. Die praktische Verwendung dieser Terminaltypen zur Jobfernverarbeitung ist vor allem durch die niedrigen Übertragungsraten begrenzt.
Eine weitere Ausbaustufe soll intelligente Terminals über SPOOLRJE an ESER-Anlagen koppeln. Sie arbeiten vorrangig mit synchroner Übertragung. Verschiedene Eingabegeräte der Terminals gestatten dem Nutzer breite Anwendungsmöglichkeiten, wie z. B. die Eingabe vorgefertigter Jobs von Magnetkassette, Floppy Disk oder Lochstreifen, Korrektur, Ergänzung bzw. Überschreiben von Jobparametern am Bildschirm. Nach dem Absenden der Jobs kann der Nutzer das Terminal zur Lösung eigener Probleme verwenden.

Die Art der am Terminal angeschlossenen Ausgabegeräte bestimmt die Ausgabemöglichkeiten der Jobergebnisse, z. B. auf dem Bildschirm, Drucker oder Kassette. Die Entscheidung über das Ziel trifft der Nutzer am Terminal oder es werden Standardannahmen vom Zentralrechner wirksam, die eine Ausgabe der Ergebnislisten auf eingeschaltete Hardcopy-Geräte vornehmen.
Weitere Terminals dieser Ausbaustufe werden fernaufgestellte Kleinrechner sein, die über Lochkarteneingabegeräte, schnelle Drucker und große externe Speichermedien verfügen. Vorgesehen ist die Einbeziehung von SKR-Rechnern und von kleineren ESER-Anlagen. Die Effektivität der Fernverarbeitung wird dann zum großen Teil durch die Übertragungsgeschwindigkeit bestimmt, d. h. es müssen Leitungen mit Übertragungsgeschwindigkeiten großer 2400 Bd verwendet werden. Die Übertragung erfolgt synchron mit speziellen Übertragungsprozeduren, die unter anderem eine Datenreduktion gestatten. Der Zentralrechner bedient mehrere Ein- und Ausgabegeräte der Terminals gleichzeitig. Über die Konsolen der Terminals und des Zentralrechners kann eine Bedienerverständigung erfolgen.

## 3. Programmtechnische Lösung

Die mit SPOOLRJE angestrebte Fernverarbeitungsvariante des SPOOL-Systems erfordert Software-Entwicklungen auf dem Zentralrechner (im SPOOL-System und im Nachrichtenkontrollprogramm), dem Verbindungsrechner (i. a. Multiplexor auf der Basis KRS 4201) und in den verschiedenen Terminalrechner (vorerst Mikrorechner auf der Basis K1510/20 bzw. R21).

### 3.1. Ergänzungen im SPOOL-System

Die Eröffnung der Leitungen erfolgt in der Initialisierungsphase des Systems oder auf Kommando des Bedieners. In einem synchron vom SPOOL-Dispatcher mit Zentraleinheitszeit versorgtem Prozessor werden zyklisch alle eröffneten Leitungen auf einen Eingabewunsch der Terminals überwacht. Liegt an einer

Standleitung ein Eingabewunsch vor, analysiert der Prozessor die Terminal- und Nutzeridentifikationsanweisung. Der Nutzer kann eine Verbindung mit dem Zentralrechner über TSO oder über SPOOLRJE wünschen. Ist das TSO nicht aktiv, werden Nutzer und Bediener davon informiert, wenn vom Terminal ein TSO-Kommando abgeschickt wurde. Soll die Eingabe über SPOOLRJE erfolgen, wird vom RJE-Prozessor, falls erforderlich, das entsprechende Nachrichtenkontrollprogramm (MCP) gestartet, das auf der Basis von TCAM die Übertragung vom Terminal zum Zentralrechner übernimmt. Das MCP wird als asynchrone Task vom RJE-Prozessor, falls es nicht schon für andere Leitungen aktiviert war, gestartet. Das SPOOL-System stellt somit für das MCP ein Anwendungsprogramm dar. Die vom MCP übernommenen Daten werden in einer Plattenwarteschlange zwischengespeichert und anschließend auf den als Internen Reader bezeichneten Eingang des SPOOL-Systems in die SPOOL-QUEUE überführt. Damit erhält SPOOL die Möglichkeit der Kontrolle und Analyse der eingegebenen Jobsteue - und SPOOL-Steuerkarten. Nachdem SPOOL anhand einer speziellen Steuerkarte das Ende des Jobstromes bzw. der Übertragung erkannt hat, wird über ein intern abgesetztes Kommando die Übertragung auf der benutzten Leitung gestoppt. Falls keine anderen Übertragungen im MCP mehr stattfinden, wird auch das MCP gestoppt. Bei Wählleitungen erfolgt zusätzlich noch ein Kennungsaustausch und eine Kontrolle der Zuschaltberechtigung des Terminals. Die eigentliche Übertragung erfolgt wie bei Standleitungen.

Die in die SPOOL-QUEUE von entfernt aufgestellten Terminals delegierten Jobs erzeugen eine Bedienerinformationsliste wie andere von entsprechenden SPOOL-Geräten eingegebene Jobs. Anhand dieser Informationen steuert der Bediener oder das System die Auswahl der abarbeitungsfähigen Jobs. Nach der Abarbeitung stehen die speziell gekennzeichneten Jobs der Fernterminals zur Ausgabe zur Verfügung. Anhand der dem Job beigefügten Informationen wird die Ausgabe am Zentralrechner oder am Terminal aktiviert.

Die Aufteilung der Ergebnisse auf die möglichen Ausgabegeräte des Terminals erfolgt durch den Nutzer bei Jobeingabe, indem

er spezielle Ausgabeklassen oder Geräte dafür vorsieht. Diese Ausgabeklassen oder Geräte beziehen sich nur auf ein bestimmtes Terminal, d. h. sie werden vom SPOOL-System nicht beachtet. Die Zuordnung zu physischen Geräten erfolgt durch das im Terminal aktivierte Programm (RTP) zur Zeit der Datenübernahme. Sollen die Ausgabedaten zum Terminal übertragen werden, versucht SPOOL nach Bestimmung des Terminals und der zugehörigen Leitung, diese zu eröffnen. Dabei muß die Empfangsbereitschaft des Terminals für mindestens ein Ausgabegerät vorliegen. Die eigentliche Übertragung übernimmt wieder die vom RJE-Prozessor gestartete MCP-Task, die nach Übertragungsende automatisch gestoppt wird, wenn keine weiteren Übertragungen vorliegen. Sollte es nicht möglich sein, die Leitung zu eröffnen oder das Terminal ist nicht bereit, so erfolgt nach einigen Versuchen eine Nachricht an den Bediener des Zentralrechners, der dann über die weitere Verfahrensweise entscheidet.
Auf diese Weise werden sowohl Stand- als auch Wählleitungen nicht für die gesamte Sitz-ungsdauer, sondern nur für die eigentliche Übertragung mit dem Rechner verbunden. Dadurch wird u. a. die Kapazität der wenigen vorhandenen Wählleitungen effektiver genutzt.
Spezielle Techniken der Datenreduktion erhöhen bei Synchronübertragung die Datenübertragungsrate und die Übertragungssicherheit.

### 3.2. Änderungen im Emulatorprogramm

Ausgehend von programmierbaren Multiplexoren machen sich Änderungen an den gebräuchlichen Emulatorprogrammen erforderlich. Die Emulatorprogramme müssen anhand spezieller Kommandocodes unterscheiden können, ob das ankommende Kommando vom SPOOL-System oder von einer anderen Zugriffsmethode ausging. Für Anforderungen von SPOOL soll das Emulatorprogramm die Kontrolle der Leitungen übernehmen und über eine Statusänderung des Subkanals dem RJE-Prozessor den Eingabewunsch des

Terminals signalisieren. Ist dagegen das MCP aktiv, so übernimmt dieses wie bisher üblich die Kontrolle der Leitungen. Für Wählleitungen muß das Emulatorprogramm so lange "Rechner besetzt" melden, bis es vom RJE-Prozessor eine Information über die Aktivierung des MCP erhalten hat. Die Delegierung der Leitungsüberwachung für die Zeit zwischen den Übertragungen an das Emulatorprogramm trägt wesentlich zur Entlastung des Multiplexkanals des Zentralrechners bei.

### 3.3. Anforderungen an das Remote Terminal Programm (RTP)

Spezielle Anforderungen an das RTP hängen sehr vom Typ und der Ausrüstung des Terminals ab. Einige grundlegende Forderungen müssen jedoch von jedem RTP erfüllt werden:
Die Eingabe der Jobs kann von verschiedenen Geräten erfolgen, wobei temporäre Modifikationen möglich sein sollen. Dabei muß sich das Terminal und der Nutzer durch spezielle Steuerkarten identifizieren (LOGON für TSO-Betrieb, TCAMON für RJE-Betrieb). Der Abschluß jedes Jobs und der gesamten Übertragung wird durch EOJ- bzw. EOF-Karten signalisiert. Terminals ohne Konsolen können über MESSAGE-Karten dem Bediener des Zentralrechners gewisse Informationen zukommen lassen. Informationen vom Bediener des Zentralrechners können dagegen in der Eingabephase nicht empfangen werden. Die Eingabedaten werden wenn möglich verdichtet übertragen. Ein Offline-Betrieb des Terminals ist nach Absenden der EOF-Karte möglich. Dabei muß jedoch von jedem offline-Programm eine primitive Leitungsüberwachung gesichert sein. Ist dies nicht möglich, so sind spezielle Zeiten zum Empfang der Daten über die Terminalidentifikation mit dem Zentralrechner zu vereinbaren. Zum Empfang der Jobausgaben muß der Nutzer gegebenenfalls die offline-Verarbeitung unterbrechen und den Empfangsteil des RTP aktivieren.

### 3.4. Auswahl des Nachrichtensteuerprogramms

Das zur Übertragung benötigte Nachrichtensteuerprogramm soll für den TSO-Betrieb und für den Anschluß anderer Anwenderprogramme geeignet sein. Deshalb sollten alle Terminals umschalt-

bar generiert werden. Die Nachrichtenbehandler müssen den TSO-Betrieb und die RJE-Komponente des SPOOL-Systems bedienen. Die Warteschlangen werden für TSO im Hauptspeicher und für SPOOL auf Wechselplatten gehalten. Die Verteilung der Nachrichten vom SPOOL-System an die entsprechenden Terminals erfolgt über die der Nachricht vorangestellten Identifikation des Terminals.

## 4. Hauptspeicher- und Direktzugriffsspeicherbedarf

Im SPOOL-System erhöht sich der Speicherbedarf bei Generierung des RJE-Prozessors auf Grund der Overlay-Struktur nur um wenige K-Byte. Während der Übertragung werden zusätzlich rund 100k Byte für das MCP benötigt, die nach dem Stoppen wieder freigegeben werden. Zusätzlicher Direktzugriffsspeicher wird nur für die Plattenwarteschlangen des MCP benötigt, die aber relativ klein gehalten werden können, da die Nachrichten bei Freiwerden eines Internen Readers sofort an das SPOOL-System übergeben werden.

## 5. Realisierungsstand

Die erste Ausbaustufe, die nur dialogorientierte Terminals, Fernschreiber an Wählleitungen und keine Veränderung am Emulatorprogramm beinhaltet, soll Mitte 1983 fertig sein. Die zweite Ausbaustufe ist vorerst für Terminals auf Mikrorechnerbasis (K1510) und für die Kopplung eines R21 an den Zentralrechner vorgesehen.

Verfasser: Dipl.-Math. Hans-Günter Meißler
Martin-Luther-Universität
Halle-Wittenberg
4020 Halle, Weinbergweg 17

Dipl.-Math. Conrad Piens, Humboldt-Universität

# Effektivitätsuntersuchungen für das System SPOOL 3

## 1. Zielstellung des Systems SPOOL 3

Das System SPOOL 3 soll aufbauend auf dem Arbeitsvorbereitungssystem SPOOL 2.1 (s. /1/) den Produktionsprozeß an ESER-Rechnern, die mit dem Betriebssystem OS/ES betrieben werden, nach folgenden Gesichtspunkten effektivieren:

- rationelle Auslastung der Magnetdatenträger (sowohl Magnetplatten als auch Magnetbänder);
- hohe Datensicherheit für den Nutzer (Entlastung des Nutzers von der Verantwortung für Sicherheitskopien seiner Dateien);
- Minimierung von Datenträgerwechseln (Magnetplatten);
- Beseitigung des Engpasses DA-Speicherplatz;
- Vergrößerung des Zeitraumes, in dem das bedienungsunabhängige Betreiben der EDV möglich ist (vgl. /2/).

Um diese Zielstellung zu erreichen, wird mit SPOOL 3 ein Konzept eines virtuellen Direktzugriffsspeichers realisiert.

Unter einem virtuellen Direktzugriffsspeicher wollen wir ein hierarchisch organisiertes Speichersystem für Dateien verstehen. Seine erste Hierarchiestufe umfaßt permanent montierte Magnetplatten, den sogenannten POOL, die zweite Hierarchiestufe ein Magnetbandarchiv. Ein Datei- und Datenträgerverwaltungssystem sorgt dafür, daß sich alle Dateien, die von Jobs benötigt werden, zu deren Abarbeitungszeit in der ersten Hierarchiestufe, also im POOL, befinden.

SPOOL 3 soll neben Dateien, die sich im virtuellen Direktzugriffsspeicher befinden, auch Dateien auf privaten Magnetplatten und die übliche Arbeit mit Magnetbanddateien zulassen.

## 2. Funktionsprinzipien von SPOOL 3

Nach dem Einlesen der Jobs erfolgt die vollständige Kontrolle der Jobsteueranweisungen auf syntaktische Korrektheit. Gleichzeitig wird die zulässige Inanspruchnahme der von den Jobs benötigten Ressourcen und die Verfügbarkeit der von ihnen benötigten Dateien im POOL geprüft. Falls nötig, werden Dateien aus der zweiten Stufe des

Speichersystems in die erste Stufe transportiert. Erst wenn alle von einem Job benötigten Dateien im POOL verfügbar sind, wird dieser Job zur Bearbeitung freigegeben. Ist im POOL kein Platz für die einzulagernden Dateien, müssen momentan nicht benötigte Dateien in die zweite Stufe ausgelagert werden.

## 3. Effektivitätsbetrachtungen

Es ist offensichtlich, daß das Konzept des virtuellen Direktzugriffsspeichers neben den gewünschten Effektivitätsgewinnen auch Belastungen für die Systemarbeit mit sich bringt. Diese Belastungen bestehen im wesentlichen in den Dateitransfers zwischen den Stufen des Speichersystems. Um also erste Aussagen über die Gesamteffektivität des Systems treffen zu können, ist die zu erwartende Häufigkeit von Dateitransfers zwischen den Stufen der Speicherhierarchie zu ermitteln. Weitere Untersuchungen beträfen z.B. die nötige Größe und mögliche Ausdehnung des Magnetbandarchivs, die zu erwartenden Häufigkeiten von notwendigen Magnetbandmontagen zur Gewährleistung des Datentransfers, die Zeit, die zur Dateibereitstellung im POOL nötig ist und die zu erwartende Häufigkeit des Verdichtens der Magnetbänder des Archivs. Möglich und notwendig sind auch Untersuchungen zur optimalen Dateiverteilung auf den Magnetplatten des POOL's (vgl. /4/).

In diesem Beitrag sollen lediglich Betrachtungen zur zu erwartenden Häufigkeit von Dateitransfers angestellt werden.

### 3.1. Modell der Datei- und Datenträgerverwaltung

Um Aussagen treffen zu können, wurde ein stark vereinfachtes Simulationsmodell der Datei- und Datenträgerverwaltungskomponenten von SPOOL 3 geschaffen (s. /3/). In diesem Modell werden beide Stufen des Speichersystems als jeweils homogenes Speichermedium aufgefaßt, das heißt, der POOL wird durch eine Magnetplatte gewünschter Kapazität und das Magnetbandarchiv durch ein Magnetband beliebiger Kapazität repräsentiert. Alle Aktionen werden zeitlos, also ohne Berücksichtigung ihrer realen Dauer, ausgeführt. Von diesem Simulationsmodell werden speziell aufbereitete SMF-Sätze vom Typ 14 und 15 (Sätze zur Erfassung der Dateiverarbeitung) verarbeitet und alle interessierenden Häufigkeiten von Anforderungen an das Datei- und

Datenträgerverwaltungssystem ermittelt. Die Größe des POOL's und in gewissem Sinne auf die Auslagerungsstrategie können variiert werden.

Bei jeder Dateianforderung wird ein Algorithmus gemäß Bild 1 abgearbeitet. Dabei erfolgt die Auslagerung von Dateien nach einer modifizierten LRU-Strategie, wobei gesteuert werden kann, ob lediglich für die zu bereitstellende Datei Platz im POOL geschaffen oder der POOL bis auf eine vorgebbare Grenze geleert werden soll.

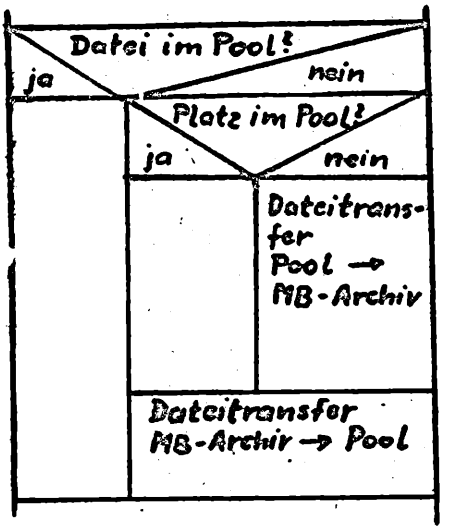



Bild 1

## 3.2. Simulationsergebnisse

Mittels des oben beschriebenen Modells wurden die von der Funktion „Erfassen der Ressourcenbenutzung" bereitgestellten Dateiverarbeitungssätze eines Zeitraumes von 61 Tagen verarbeitet. In diesem Zeitraum wurde von 3556 Jobs zu 444 Dateien, die der Datei- und Datenträgerverwaltung von SPOOL 3 unterliegen sollen, insgesamt 19054 mal zugegriffen. Diese Dateien belegen einen Speicherplatz von 39248 Spuren auf WPS vom Typ 5061, das entspricht also einer Kapazität von ca. 290 M Byte. In den verschiedenen Simulationsläufen wurden die POOL-Kapazität und der bei Auslagerungen frei zu machende Speicherplatz variiert, um Zusammenhänge zwischen diesen Größen und den Dateitransfers zwischen POOL und Magnetbandarchiv ableiten zu können. Bei allen hier dargestellten Ergebnissen wird

davon ausgegangen, daß sich zum Beginn des Simulationszeitraumes alle Dateien ausschließlich im MB-Archiv befinden. Andere Simulationsläufe haben ergeben, daß eine Vorbelegung des POOL's nur unwesentlichen Einfluß auf die Anzahl der nötigen Dateitransfers hat.

In der Tabelle 1 bezeichnen

- P - POOL-Kapazität in Spuren auf WPS 5061
- FP - frei zu machenden Speicherplatz in % der POOL-Kapazität, Ø bedeutet, daß nur für die anliegenden Dateieinlagerungen Platz geschaffen wird
- $\overline{J}$ - arithmetisches Mittel der Anzahl von Jobs pro Tag, die Dateieinlagerungen erfordern
- $\overline{D}$ - arithmetisches Mittel der Tage, die zwischen zwei notwendigen Auslagerungen liegen
- PJ - Anteil der Jobs, die Dateieinlagerungen verlangen, in Prozent
- $\overline{DT}$ - arithmetisches Mittel der Dateitransfers pro Tag
- $\overline{DE}$ - arithmetisches Mittel der Dateieinlagerungen pro Tag.

| P | FP | $\overline{J}$ | $\overline{D}$ | $\overline{DT}$ | $\overline{DE}$ | PJ |
|---|---|---|---|---|---|---|
| 16000 | 0 | 11,5 | 0,1 | 36,2 | 23,7 | 21,1 |
| | 10 | 12,5 | 0,7 | 40,3 | 25,2 | 21,4 |
| | 20 | 13,8 | 1,2 | 44,0 | 27,6 | 23,7 |
| | 30 | 14,4 | 1,6 | 47,1 | 29,5 | 24,8 |
| | 40 | 15,2 | 1,9 | 49,7 | 31,0 | 26,0 |
| 20000 | 0 | 7,1 | 0,2 | 20,8 | 14,1 | 12,2 |
| | 10 | 7,5 | 1,9 | 22,3 | 14,9 | 12,9 |
| | 20 | 8,1 | 3,3 | 23,9 | 16,7 | 13,9 |
| | 30 | 9,0 | 4,0 | 27,9 | 18,3 | 15,3 |
| | 40 | 10,0 | 4,4 | 31,9 | 20,7 | 17,2 |
| 24000 | 0 | 5,3 | 0,4 | 14,4 | 10,0 | 9,1 |
| | 10 | 5,5 | 4,4 | 14,9 | 10,8 | 9,5 |
| | 20 | 6,1 | 7,0 | 16,5 | 11,7 | 10,4 |
| | 30 | 6,4 | 8,3 | 17,4 | 12,2 | 10,7 |
| | 40 | 6,3 | 9,7 | 20,5 | 13,8 | 11,7 |

Tabelle 1

Die in der Tabelle 1 dargestellten Simulationsergebnisse zeigen den erwarteten Zusammenhang zwischen POOL-Größe und Anzahl der Dateitransfers zwischen den Hierarchiestufen des Speichersystems, nämlich daß mit wachsender POOL-Größe die Anzahl der Dateitransfers abnimmt. Es zeigt sich, daß diese Zahl kleiner wird, wenn bei notwendigen Auslagerungen weniger Platz im POOL geschaffen wird. (Die Angabe 0 in der Tabelle sagt aus, daß nur soviel Platz im POOL geschaffen wird, wie für die einzulagernde Datei benötigt wird).

Abschließend ist zu bemerken, daß die bisherigen Ergebnisse lediglich Tendenzen erkennen lassen. Die Zahl der bisher durchgeführten Simulationen ist zu gering, um die in der Tabelle 1 dargestellten Ergebnisse als statistisch gesichert zu bezeichnen.

Literatur:

/1/ Winks, Hans: SPOOL - ein System zur Automatisierung von Produktionsprozessen an ESER-Rechnern

/2/ Kroß, Günther: Studie zur Weiterentwicklung des Systems SPOOL hinsichtlich der Automatisierung und Rationslisierung von Produktionsprozessen an ESER-Rechnern (SPOOL 3), ORZ der HUB 1981, unveröffentlichtes Arbeitspapier

/3/ Schöbel, Christiane: Simulation der Dateibereitstellungskomponenten des Systems SPOOL 3, Fachschule für Ökonomie Rodewisch, Außenstelle Berlin 1982

/4/ Herzholz, Gerhard: Leistungsbewertung im Rechenbetrieb in diesem Heft

Weickmann, Christoph

# Gemeinsame Nutzung von TSO und SPOOL auf einer EDVA

## 1. Ausgangspunkt

Das Teilnehmersystem TSO bietet u.a. die Möglichkeit, Stapeljobs im Dialog - z.B. mit Hilfe des Kommandos EDIT - zu erstellen oder zu aktualisieren und diese so behandelten Jobs direkt in die Jobkettendatei des OS/ES zu delegieren. Diese Art der Arbeit hat den Vorteil, daß rechnerintensive Aufgaben nicht den Dialog belasten, sondern parallel zu TSO oder anschließend abgearbeitet werden. Dies gilt z.B. für längere Übersetzungsläufe, Jobs der Produktionssteuerung oder auch Aufgaben des TSO-Dialogs für das PP Statistik. Der Teilnehmer nutzt dadurch TSO mehr für die Programmerstellung und -testung sowie für die Lösung von Aufgaben, die dialogorientiert aufgebaut sind.

Diese Möglichkeit, den Vordergrund (Dialogarbeit) mit dem Hintergrund (Stapelarbeit) zu verbinden, wird Foreground-Initiated-Background (FIB) genannt. Für FIB gibt es eine Gruppe von Kommandos, für die der Teilnehmer vom Rechenzentrum privilegiert sein muß. Sie ermöglichen:

- Delegieren von Jobs;
- Streichen von delegierten Jobs;
- Ausgabe der Ergebnisse von delegierten Jobs am Terminal;
- Anzeige des Bearbeitungsstandes delegierter Jobs am Terminal.

Es handelt sich hierbei um die Kommandos SUBMIT, CANCEL, OUTPUT und STATUS.

### Aufgaben der Kommandos

SUBMIT:

- Prüfen, ob der Teilnehmer für FIB zugelassen ist;
- Hinzufügen einer Jobkarte, falls der Jobstrom nicht mit einer beginnt;
- Hinzufügen eines Unterscheidungszeichens, falls der Jobname aus dem Teilnehmerkennzeichen besteht;
- Möglichkeiten eines Exits, mit dessen Hilfe das Rechenzentrum technologische Vorgaben wie CPU-Zeit, Regionsgröße, CLASS-Par

meter oder SYSOUT-Klassen automatisch prüfen oder ergänzen kann, bzw. durch den Job auch gestrichen werden können, wenn sie diese Vorgaben nicht enthalten;
- Delegieren des Jobs in die Jobkettendatei des OS/ES.

CANCEL:
- Prüfen, ob der Teilnehmer für FIB zugelassen ist;
- Prüfen, ob der Jobname mit dem Teilnehmerkennzeichen beginnt;
- Streichen des Jobs aus einer der Ketten der Jobkettendatei bzw. aus der Abarbeitung.

OUTPUT:
- Prüfen, ob der Teilnehmer für FIB zugelassen ist;
- Prüfen, ob der Jobname mit dem Teilnehmerkennzeichen beginnt;
- Ausgabe von SYSOUT-Dateien am Terminal;
- Umketten von SYSOUT-Dateien in Klassen, die vom Systemausgabeprogramm bedient werden;
- Streichen nicht benötigter SYSOUT-Dateien.

STATUS:
- Prüfen, ob der Teilnehmer für FIB zugelassen ist;
- Anzeige des Bearbeitungsstandes der Jobs am Terminal.

Verbindung zu SPOOL

Da das Preschedulingsystem SPOOL an Rechenzentren des Hochschulwesens immer häufiger zum Einsatz kommt, ist es wichtig, die von TSO delegierten Jobs dem technologischen Regime von SPOOL anzupassen. Das heißt, daß diese Jobs auch durch die Jobauswahl, die Druckaufbereitung, die Prüfungen zur Verfügbarkeit von Ressourcen usw. von SPOOL bedient werden. Andererseits würde es unnötige technologische Probleme geben, wenn bei paralleler Arbeit von TSO und SPOOL Jobs sowohl direkt - mittels TSO-Kommando - als auch über die Jobauswahl von SPOOL in die Jobkettendatei können. Damit würden auch alle technologischen Vorteile verloren gehen, die in den Jobklassifizierungen, der Prüfung der Ressourcennutzung, der Möglichkeit des bedienungsunabhängigen Betriebes der Druckunterstützung usw. liegen.

Auch um eine Vereinheitlichung in der Stapelarbeit zu erreichen, werden die nachfolgend beschriebenen Änderungen an den FIB-Kommandos vorgenommen.

## 2. Aspekte der gemeinsamen Nutzung von TSO und SPOOL auf einer EDVA

Bei der gemeinsamen Nutzung von TSO und SPOOL muß der zur Verfügung stehende Hauptspeicher berücksichtigt werden. Steht eine schnelle Anlage von ein MB und mehr zur Verfügung, kann eine parallele Arbeit von TSO und SPOOL e ngeplant werden. Da es jedoch auch in Zukunft Anlagen mit 512 K Hauptspeicher geben wird und auf diesen TSO und SPOOL sinnvoll genutzt werden wird, muß die Möglichkeit einer seriellen Arbeit von Dialog und Stapel berücksichtigt werden.

Entsprechend diesen zwei Aspekten - parallele bzw. serielle Arbeit von Dialog und Stapel - wird bei der Bearbeitung der Kommandoprozessoren unterschieden in

a) SPOOL ist im System und aktiv,
b) SPOOL ist im System und nicht aktiv.

## 3. Leistungen der geänderten Kommandoprozessoren

a) SPOOL ist im System und aktiv

Generell gilt für alle Kommandoprozessoren, daß die Leistungen - insbesondere die technologische Seite betreffend - voll erhalten bleiben.

SUBMIT:

Die Jobs werden in die SPOOL-Jobdatei (SPOOLQUEUE) delegiert. Dafür wird das Interface des internen Readers von SPOOL benutzt.

Um dieses Interface nutzen zu können, ist es erforderlich, umfangreiche Anpassungen an den Routinen zur dynamischen Dateizuweisung (DAIR) für die TSO-Arbeit vorzunehmen.

Die dynamische Dateizuweisung unter TSO ermöglicht es, daß sich der Teilnehmer während einer Sitzung Dateien nach Bedarf zuweisen oder freigeben kann. Dies wird unter TSO jedoch nur für Dateien unterstützt, die sich auf DA-Geräten befinden. Magnetbandgeräte, Drucker, Lochkartenleser usw. werden nicht prinzipiell unterstützt, da hierbei vom Teilnehmer nicht nur eine Datei, sondern ein ganzes Gerät blockiert würde.

Um jedoch für SUBMIT das Interface des internen Readers von SPOOL zu nutzen, ist es notwendig, daß dem Teilnehmer ein SPOOL-

DUMMY-Gerät dynamisch zugewiesen und von ihm später wieder freigegeben wird.

CANCEL:

Für die Ausführung dieses Kommandos werden die Möglichkeiten von SPOOL genutzt, indem die Steuerblöcke von SPOOL durchsucht werden. Das Streichen eines gefundenen Jobs erfolgt dann mittels eines SPOOL-Kommandos.

OUTPUT:

Der Teilnehmer muß, wenn er sich SYSOUT-Dateien über das Terminal ansehen will, für seine zu delegierenden Jobs SYSOUT-Klassen wählen, die in der Jobkettendatei verbleiben. Für das Umketten von SYSOUT-Dateien werden Klassen vorgemerkt, die außerhalb von SPOOL ausgedruckt werden.

STATUS:

Informationen über den Bearbeitungsstand eines Jobs werden den Steuerblöcken von SPOOL entnommen.

b) SPOOL ist im System und nicht aktiv

Auch hierbei gilt, daß der Leistungsumfang der Kommandoprozessoren nach Möglichkeit erhalten bleibt.

SUBMIT:

Die Delegierung von Jobs erfolgt hier in eine untergliederte Datei. Dabei wird als Bestandsname der Jobname gewählt. Existiert ein Bestandsname schon, so wird ein neuer Jobname angefordert. Die Jobs aus dieser Datei werden beim SPOOL-Start in die SPOOL-Jobdatei übernommen, und nach erfolgreicher Übernahme wird der bisher belegte Platz freigegeben.

CANCEL:

Das Streichen von Jobs wird sich hierbei nur auf die untergliederte Datei beschränken.

OUTPUT:

Für die Möglichkeiten von OUTPUT bei der seriellen Arbeit von Dialog und Stapel gilt das Gleiche wie bei der parallelen Arbeit, obwohl die Anwendung dieses Kommandos infolge der zeitlichen Verschiebung von Dialog-Stapel-Dialog nicht sehr sinnvoll ist.

STATUS:

Delegierte Jobs werden nur in der PO-Datei gesucht.

## 4. Bearbeitung

Die Änderungen an den Kommandoprozessoren erfolgen an eindeutigen Schnittstellen mit Hilfe von Rahmenprogrammen, so daß Änderungen des Systementwicklers ohne weitere Probleme wirksam werden können. Die vorhandenen Routinen der FIB-Kommandos werden weitestgehend genutzt.

Verfasser:

Dipl.-Math. Christoph Weickmann
Humboldt-Universität
ORZ
1086 Berlin, PSF 1297

Wörfel, Friedhard

# Aufbau und Handhabung des Systems der Katalogdateien für SPOOL

## 1. Einleitung

Das SPOOL-System ist ein Pre- und Post-Schedulingsystem, das als ergänzende Komponente des Betriebssystems OS/ES für die Steuerprogrammkonfiguration MFT und MVT einsetzbar ist. SPOOL übernimmt in effektiver Form die Jobeingabe und -ausgabe, führt Prüfungen zur Zugriffsberechtigung zu Dateien bzw. Datenträgern, sowie zur Verfügbarkeit der Ressourcen Rechnerlaufzeit (Rechnerkosten), Druckerpapier und Speicherplatz auf Magnetplatte aus.

## 2. Das Syster der Katalogdateien

Die Katalogdateien des SPOOL-Systems stellen eine Gruppe miteinander verketteter Dateien dar, die zur Zugriffsprüfung und zur Verbrauchserfassung von bestimmten Ressourcen dienen, die von den im SPOOL-System bearbeiteten Jobs benutzt werden. Folgende Katalogdateien sind im SPOOL-System vorhanden.,

### 2.1. Auftragsnummerndatei

Diese Datei ist ein Verzeichnis der im SPOOL-System registrierten Aufträge. Jeder Auftrag wird durch eine 8-stellige Auftragsnummer gekennzeichnet. Zu jeder Auftragsnummer sind Informationen über Beginn und Ende der Auftragsgültigkeit und über die vom Auftrag kummultativ in Anspruch genommenen Größen des Papierverbrauchs, des Direktzugriffsspeicherplatzes und der Rechnerkosten vorhanden. Für die Verbrauchsgrößen sind Limitwerte eingetragen, die nicht überschritten werden dürfen. Zu jeder Auftragsnummer können beliebig viele 8-stellige Nutzerkennzeichen vergeben werden. In diesem Fall werden auch die

Verbrauchsgrößen für die einzelnen Nutzerkennzeichen erfaßt. Eine Limitierung der Verbrauchsgrößen ist nur für den gesamten Auftrag möglich. Die inhaltliche Bedeutung des Nutzerkennzeichens bleibt dem jeweiligen Anwender des SPOOL-Systems überlassen. Auftragsnummer und Nutzerkennzeichen werden aus geeigneten Parametern der Jobanweisung entnommen.

2.2. Dateiverzeichnis

Dieses Verzeichnis enthält Informationen zu allen Dateien, die sich auf Direktzugriffsdatenträgern befinden und für eine permanente Nutzung durch unterschiedliche Jobs im SPOOL-System zugänglich sein sollen. Als Dateinamen sind im SPOOL-System nur einfache oder untergliederte Namen zugelassen. Zwei physisch verschiedene Dateien dürfen nicht den gleichen Namen besitzen. Zu jedem Dateinamen muß ein autorisierter Auftrag durch seine Auftragsnummer angegeben werden. Als Zusatzinformation kann ein zugehöriges Nutzerkennzeichen eingetragen werden. Registriert werden das Einrichtungsdatum, das Datum des letzten Zugriffs, die Zahl der Zugriffe, das Verfallsdatum und die Primärspeicherplatzmenge. Der Datenträger auf dem sich die Datei befindet, wird durch einen Verweis auf den Satz des Datenträgerverzeichnisses festgehalten. Alle Dateien auf dem gleichen Datenträger sind durch die entsprechenden Verweise auf die Sätze des Dateiverzeichnisses miteinander vorwärts und rückwärts verkettet. Das gleiche gilt für Dateien, die in der höchsten Indexstufe den gleichen Namen besitzen.

2.3. Datenträgerverzeichnis

Diese Datei enthält Angaben zu jedem Datenträger, der im SPOOL-System registriert ist. Zu jedem Datenträgernamen muß ein autorisierter Auftrag durch seine Auftragsnummer angegeben sein. Als Zusatzinformation kann ein zugehöriges Nutzerkennzeichen eingetragen werden. Außerdem werden eine Eigentümeridentifikation und der Gerätetypcode eingetragen. Bei Direktzugriffsdatenträgern wird ferner die Anzahl der (im Rahmen des SPOOL-

Systems) angelegten Dateien und zwei Verweise in das Dateiverzeichnis auf den Anfang und das Ende der Kette der Dateien dieses Datenträgers abgespeichert. Alle Datenträger, deren Namen mit der gleichen 3-stelligen Zeichenkette beginnen, werden durch Verweise auf die entsprechenden Sätze des Datenträgerverzeichnisses vorwärts und rückwärts verkettet.

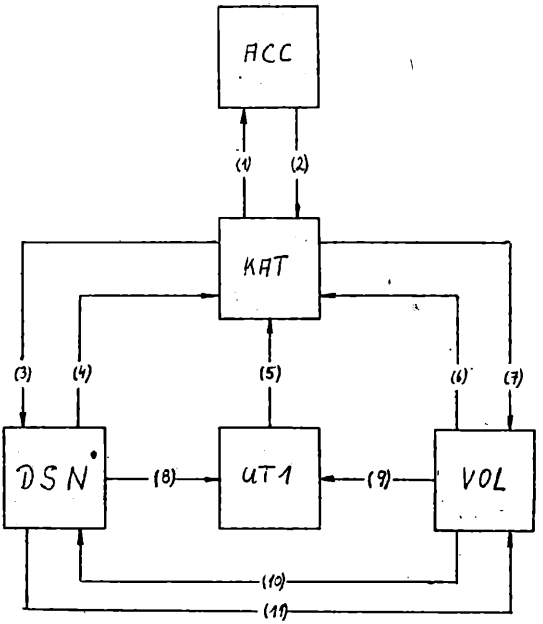
## 2.4. Zentraler Katalog

Der zentrale Katalog ist eine nach den Auftragsnummern gegliederte Hilfsdatei. Sie ermöglicht einen einfachen und schnellen Zugriff zu den bereits genannten Verzeichnissen. Es sind Informationen abgespeichert, die die Verfügbarkeit der einzelnen Aufträge über Dateien und Datenträger regeln.

## 2.5. Rückpointerdatei

Diese Datei ist eine weitere Hilfsdatei. Sie hat die Aufgabe, eine Rückverkettung vom Dateiverzeichnis und Datenträgerverzeichnis zu allen Aufträgen zu ermöglichen, die eine Verfügbarkeit über eine bestimmte Datei oder einen bestimmten Datenträger besitzen. Es wird nur die Verfügbarkeit aller nichtautorisierten Aufträge geregelt. Alle autorisierten Aufträge sind direkt über den Zentralen Katalog mit ihren zugeordneten Dateien oder Datenträgern verkettet.

2.6. Schema der Verkettung




Erläuterungen:

KAT - Zentraler Katalog
ACC - Auftragsnummerndatei
DSV - Dateiverzeichnis
VOL - Datenträgerverzeichnis
UT1 - Rückpointerdatei

(1 ) - Verweis auf einen Satz
(2 ) - Auftragsnummer
(3 ) - Verweis auf einen Satz
(4 ) - Autorisierte Auftragsnummer
(5 ) - Auftragsnummer
(6 ) - Autorisierte Auftragsnummer
(7 ) - Verweis auf einen Satz
(8 ) - Verweis auf einen Satz

(9 ) - Verweis auf einen Satz
(10) - Verweis auf zwei Sätze
(11) - Verweis auf einen Satz

## 3. Der interne Aufbau der Katalogdateien

Alle Katalogdateien des SPOOL-Systems sind nach einem einheitlichen Prinzip aufgebaut. Außer dem zentralen Katalog, der eine untergliederte Datei ist, sind alle anderen Dateien sequentiell organisiert. Sämtliche Dateien enthalten ungeblockte Sätze fester Länge. Der gesamte Datenbereich der Dateien ist formatisiert und die Sätze sind in mehrfacher Weise untereinander verkettet. Die freien Sätze jeder Datei bilden die sogenannte Freibereichskette. Durch Ein- und Ausketten von Sätzen der Freibereichskette ist es möglich, den freien Bereich der Dateien wartungsfrei zu verwalten. Eine Reorganisation der Dateien ist nur dann erforderlich, wenn nicht mehr genügend freie Sätze vorhanden sind.

Zur Verkettung der Sätze werden die relativen Spur- und Satzadressen in den jeweiligen Dateien benutzt. Diese sogenannten TTR-Pointer werden immer in einem an Wortgrenze ausgerichteten Bereich von vier Byte Länge eingetragen. Die rechten drei Byte enthalten die relative Spur- und Satzadresse des Satzes, auf den verwiesen wird. Im linken Byte ist das Bit Ø für eine Anzeigefunktion reserviert. Die Bits 1 - 7 können für spezielle Kennzeichnungen verwendet werden. Das erste Wort der Sätze aller Katalogdateien enthält stets einen TTR Pointer, durch den primär zusammengehörende Sätze einer Datei verkettet sind. Im linken Byte dieses Wortes wird in den Bits 1 - 3 die Art des Satzes und in den Bits 4 - 7 die Anzahl der Eintragungen im Satz angezeigt.

## 4. Das Katalogdatei-Dienstprogramm IEBSPOOL

Aufgabe des Programms IEBSPOOL ist es, das Eintragen, Ändern oder Entfernen von Informationen in bestehenden Katalogdateien auszuführen. Die einzelnen Funktionen werden im allgemeinen über Steueranweisungen angefordert. Das Aufnehmen und Entfer-

nen von Dateien erfolgt automatisch, wenn dem Jobschritt für IEBSPOOL entsprechende DD-Anweisungen beigefügt werden. Einem bestimmten Job ist der Zugriff und die Veränderung nur an solchen Informationen der Katalogdateien gestattet, für die er durch seine Auftragsnummer autorisiert ist. Ausgenommen von dieser Regel sind sogenannte privilegierte Jobs. Privilegierte Jobs können jede beliebige Änderung am System der Katalogdateien vornehmen.

## 4.1. Aufbau der Steueranweisungen für IEBSPOOL

Die Steueranweisungen besitzen Lochkartenformat. In Spalte 1 steht das Kartenkennzeichen $, der Anweisungstext steht in den Spalten 2 - 71. Die Spalten 72 - 80 sind bedeutungslos. Der Anweisungstext hat folgenden Aufbau

$ [Name] OPERATIONSCODE OPERANDEN [Kommentar]

Die einzelnen Felder werden durch mindestens ein Leerzeichen voneinander getrennt. Das Operandenfeld besteht aus einem Stellungsoperanden und aus verschiedenen Kennwortoperanden. Eine Fortsetzung des Operandenfeldes auf Folgekarten ist möglich und wird durch ein Komma mit einem darauffolgenden Leerzeichen angezeigt. Auf der Fortsetzungskarte muß in Spalte 1 das Kartenkennzeichen stehen.

## 4.2. Operationscode

Durch den Operationscode wird im allgemeinen eine Gruppe von Funktionen des Programms ausgewählt. Die genaue Funktion wird erst durch die Angabe bestimmter Kennwortoperanden festgelegt. Es gibt folgende Operationscode:

| | | |
|---|---|---|
| PARM | - | Steuern des Programmlaufes |
| OPEN | - | Hinzufügen eines neuen Auftrages |
| CLOSE | - | Stornieren eines Auftrages |
| PURGE | - | Entfernen eines Auftrages |
| ADD | - | Hinzufügen von Informationen in die Dateien |

CHANGE - Ändern von Informationen in den Dateien
DELETE - Entfernen von Informationen aus den Dateien
CONNECT - Herstellen einer Beziehung zwischen Objekten
RELEASE - Aufheben einer Beziehung zwischen Objekten
RENAME - Umbenennen von Objekten

4.3. Stellungsoperand

Alle Steueranweisungen besitzen einen Stellungsoperanden, dessen Wert stets die Nummer eines autorisierten Auftrages darstellt.

4.4. Kennwortoperanden

In den Steueranweisungen kommen 21 verschiedene Kennwortoperanden vor. Die gewünschte Programmfunktion wird jeweils durch eine Kombination eines Operationscodes mit bestimmten Kennwortoperanden ausgewählt.

## 5. Das Initialisierungsprogramm SPLINIT

Aufgabe des Programms SPLINIT ist es, das Initialisieren der SPOOL Katalogdateien vorzunehmen. Dazu wird der gesamte Primärdatenbereich der entsprechenden Dateien im OUTPUT-Modus formatisiert. Bei der untergliederten Datei wird außerdem das Verzeichnis geleert. Alle Datensätze werden als frei gekennzeichnet und miteinander verkettet.

## 6. Das Korrekturprogramm SPLFILL

Aufgabe des Programms SPLFILL ist:

- füllen der initialisierten Katalogdateien mit Testdaten,
- korrigieren von Satzteilen in hexadezimaler und in Zeichenkettenform,
- erzeugen von Bestandseintragungen im Zentralen Katalog,
- korrigieren des Nutzerdatenfeldes von Bestandseintragungen im Zentralen Katalog.

## 7. Das Druckprogramm SPLLIST

Aufgabe des Druckprogramms SPLLIST ist das Auflisten von objektbezogenen Informationen und das Auflisten von dateibezogenen Informationen.

## 8. Das Wiederherstellungsprogramm SPLREST

Aufgabe des Programms SPLREST ist das Wiederherstellen aller Dateien im Fehlerfall mittels Sicherungsabzug und LOG-Datei.

## 9. Das Reorganisationsprogramm SPLCOPY

Aufgabe des Programms SPLCOPY ist die Erweiterung der Freibereichskette einzelner Dateien und die Reorganisation aller Dateien.

## Literaturhinweise

/1/ SPOOL-Nutzerhandbuch - Version 2.0,
Humboldt-Universität Berlin, Berlin 82

/2/ Dienstprogramm IEBSPOOL, interne Mitteilung der
Martin-Luther-Universität Halle, Halle 82

Verfasser: Dr. rer. nat. Friedhard Wörfel
Martin-Luther-Universität
Halle-Wittenberg
Organisations- und Rechenzentrum
4020 Halle
Weinbergweg 17

Anke, Hartmut

# Erfahrungen mit einem System zur Jobverwaltung

## 1. Einleitung

In einem Rechenzentrum mit massenhaftem Anfall von Aufträgen ist die Jobverwaltung ein permanentes Problem. Die Lagerung, das Wiederauffinden oder der Tranport von Lochkarten innerhalb und außerhalb des RZ, die Zuordnung von Ausgangs- und Ergebnisdaten, all das sind Prozesse mit hohem manuellen Aufwand und sich ständig wiederholenden Routinehandlungen. Darüberhinaus sind die Jobdurchlaufstrategien selbst vor allem durch ihre manuelle Realisierung geprägt. Hier spielen weniger volkswirtschaftliche Aspekte eine Rolle, als vielmehr ihre Machbarkeit bezüglich des Aufwandes.
Ausgehend von der Verpflichtung durch Forschungs- und Entwicklungsarbeiten ökonomische Effektivität zu gewinnen, versucht das nachfolgend beschriebene Programmsystem einige Probleme zu lösen bzw. deren Lösung zu unterstützen. Hierbei geht es um die Einsparung lebendiger Arbeit, die Intensivierung aller Prozesse vor allem durch Erhöhung des Automatisationsgrades und die intensive Ausnutzung der vorhandenen Technik, aber auch um die Nutzung des vorhandenen wissenschaftlichen Potentials, vorhandene Erkenntnisse praxiswirksam zu machen.
Geleitet von dieser grundsätzlichen Einstellung wurde im Rechenzentrum der Technischen Universität Dresden ein Preschedulingsystem entwickelt, implementiert und in der Produktion erprobt, das Auftragswarteschlangen verwaltet. Wir bezeichnen dieses Programmsystem als Job-Queue-Management-System (JQMS).

## 2. Entwicklungsziele

Das JQMS ist grundsätzlich zur Unterstützung eines stapelverarbeitenden Betriebssystems gedacht. Das entspricht auch der vorherrschenden Betriebsweise des Rechnersystems BESM6/ES1020 am RZ der TUD, welches sich als möglicher erster Einsatzfall anbietet.
Folgende allgemeingültige Entwicklungsziele wurden festgelegt:

- Unterstützung eines prioritätsgesteuerten Jobdurchlaufs

Unter Jobdurchlauf verstehen wir die Gesamtheit aller Prozesse im Rechenzentrum, die die Verweilzeit einer Aufgabe beeinflussen, beginnend bei der Jobabgabe und endend mit der Rückgabe der Ergebnisse. Als dominierende Einflußgröße soll dabei die Dringlichkeit der Aufgabe wirken.

- Bereitstellung effektiver Zugriffsmöglichkeiten zu Jobwarteschlangen

In diesem Zusammenhang verstehen wir unter Effektivität die Tatsache, daß der Maschinenbediener aus einer aktuellen Situation im Produktionsbetrieb heraus - einfach, schnell, flexibel - ganze Jobstapel erzeugen kann, die er zur Abarbeitung benötigt. Dabei soll ihm ein bestimmter - aber besonders im Verhältnis zur Priorität der Dringlichkeit wohldefinierter - Freiraum zur Verfügung stehen.

- Unterstützung des rationellen Umgangs mit Rechnerressourcen

Unter der Voraussetzung einer Ressourcenzuteilung können eine laufende, automatische Kontrolle und daraus abgeleitete Maßnahmen zur Steuerung des Nutzerverhaltens beitragen. Die Überwachung bezieht sich dabei sowohl auf den Ressourcenverbrauch selbst, als auch auf die zeitliche Inanspruchnahme.

- Unterstützung für Leitungs- und Planungsprozesse im Rechenzentrum

Dazu sollen vorrangig objektive Daten zur Verfügung gestellt werden, die bei der Beurteilung von Prozessen oder Technologien mehr und mehr die subjektiven Einschätzungen durch objektiv begründete Faktoren ersetzen. Die Transparenz des Gesamtablaufes soll durch geeignet verfügbare Meßdaten erhöht

werden. Die Planung des durchgehenden Schichtbetriebes sowie die Erarbeitung von Prioritätsstrategien soll unterstützt werden.

3. Systemarchitektur

Die o.g. Zielstellung und das Bestreben, die Abbildung des unterstützten Betriebssystems streng von den allgemeingültigen Algorithmen zu trennen, führten zu einem 2-Schichtmodell.

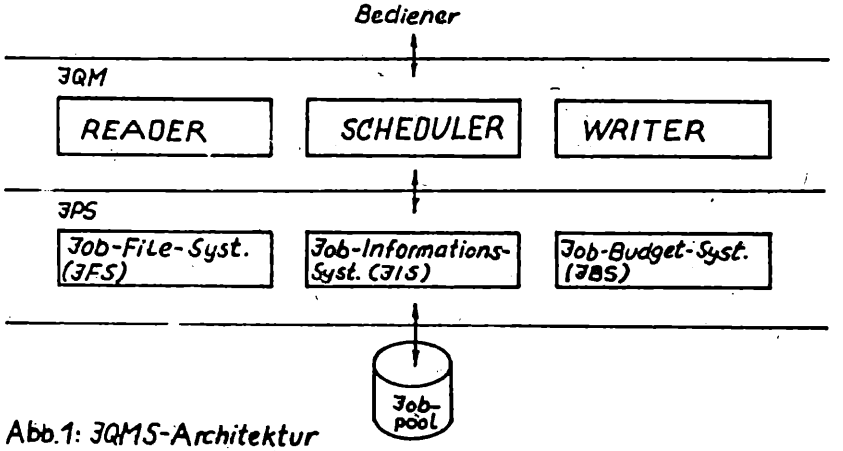



Abb.1: JQMS-Architektur

Die Grobstruktur des JQMS (Abb.1) zeigt die strikte Trennung der Management-Ebene (JQM) von der Datenverwaltung (JPS). Im JQM liegt die Jobdurchlaufstrategie, einschließlich der Kommunikation mit dem Bediener, also der auf den Anwendungsfall zugeschnittene Teil des JQMS, während das JPS als allgemeingültigen Komplex die Datenstrukturen, Zugriffspfade und die Speicherverwaltung der Datenbasis enthält.
Ein wichtiges Merkmal der Architektur ist die weitgehende Unabhängigkeit der beiden Schichten voneinander. So ist es ohne weiteres möglich, die Jobverwaltung im JQM zu verändern, ohne Auswirkungen auf Programme im JPS. Optimierungen der Zugriffspfade sind umgekehrt ohne Einfluß auf die Programme möglich, die den Jobdurchlauf realisieren. Die Schnittstelle selbst ist ein

Satz leistungsfähiger Operatoren, der alle denkbaren Manipulationen mit den relevanten Elementen erlaubt.
Die Datenwiedergewinnung ist im JIS konzentriert. Auf der Grundlage des Codd'schen relationalen Datenkonzepts wird eine Inhaltswiedergewinnung mit teilweiser Invertierung durchgeführt [1]. Das Job-File-System verwirklicht die Datenstruktur des Pools und verwaltet mit einer Belegungstabelle den Speicherplatz der Datenbasis. Die physische Speicherung ist zufällig. Transferblöcke werden bei Bedarf gekettet. Im JBS befinden sich die Algorithmen zur Überwachung der Nutzerbudgets.

## 4. Realisierung eines JQMS

Die Installation des JQMS am RZ der TUD zeigt Abb.2.

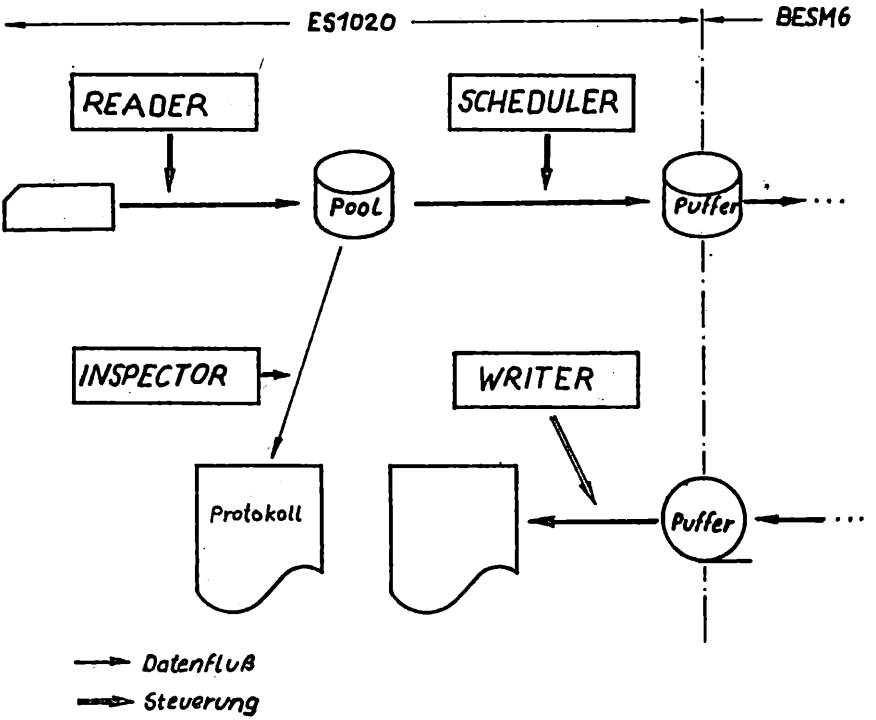



Abb.2: JQMS-Installation

Die einzelnen Systemkomponenten haben folgende Funktionen:

READER

- Übernahme der primären Jobdaten von Lochkarte auf den Pool
- Einreihen der Jobs in die entsprechende Auftragswarteschlange
- Interpretieren spezieller Jobdaten, Erzeugen von Sekundärdaten zur Unterstützung der Suche
- syntaktische Kontrolle des Jobs entsprechend der Jobsteuersprache BAMOS
- Realisierung eines Packungsalgorithmus zwecks ökonomischer Verwendung von Plattenspeicherplatz
- Mitteilungen an den Bediener

SCHEDULER

- Bereitstellen von Jobmengen in Kommunikation mit dem Bediener
- flexible Manipulation der ausgewählten Menge entsprechend der Situation am Rechnersystem
- Bereitstellen der endgültigen Jobstapel für den Hochleistungsrechner

WRITER

- Ausgabe der Jobausgabeinformation

INSPECTOR

- Ausgabe eines Inspektionsprotokolls auf Anforderung durch den Bediener
- Realisierung eines Meßapparates zur automatischen Datensammlung

Über diese Komponenten hinaus werden noch realisiert:

TERMINATOR

- Beenden einer laufenden Komponente des JQMS

RESTART

- Wiederanlauf bzw. Neuanlegen eines Jobpools nach Ausnahmesituationen am Trägerrechner (Maschinenfehler, Wartung o.ä.)

## 5. Produktionstest

Das JQMS wurde schrittweise unter Produktionsbedingungen am Rechnersystem BESM6/ES1020 getestet. Trägerrechner war die EDVA ES1020 unter Steuerung des Plattenbetriebssystems DOS. Verwaltet wurden Jobs für das Betriebssystem BAMOS des sowjetischen Hochleistungsrechners BESM6. Abschluß dieser Phase war ein einwöchiger Produktionsbetrieb, bei dem der gesamte Auftragsanfall (ca. 1600 Jobs, 300 000 Lochkarten) unter Beibehaltung der derzeit manuell realisierten Jobdurchlaufstrategien bearbeitet wurde.
Ziel der praktischen Erprobung war es, Aussagen zu den Problemkreisen
- Systemeigenschaften des JQMS
- Gesamtverhalten ES1020/JQMS
- Meßdaten über Bediener- und Benutzerverhalten

zu erhalten.

Nach den Produktionstests kann allgemein eingeschätzt werden, daß die Funktionsfähigkeit des JQMS nachgewiesen ist. Die Handhabung des Systems ist in kurzer Zeit vom Bediener zu erlernen. Die Robustheit gegenüber der Bedienung und bezüglich des Zustandes des Trägerrechners ist ausreichend.

Als sehr flexibel erwies sich das JQMS gegenüber vorher nicht erkannten Anforderungen, die sich aus dem praktischen Einsatz unmittelbar ergaben. Auf Grund der Systemarchitektur (Abb.1) war es möglich, mit geringem Zeitaufwand die Auswahlmöglichkeiten im Scheduler zu erweitern bzw. den aktuellen Verhältnissen im Jobpool anzupassen und Änderungen in der Datenstruktur von einem Tag zum anderen durchzuführen.
Die Datenreproduzierbarkeit und Datenintegrität beschränkt sich in der gegenwärtigen Systemversion auf die Komponente RESTART. Aus einer Jobrepräsentation wird eine fehlerfreie Struktur der Verwaltungstabellen wiederhergestellt. Die

Die Eigenschaften der Komponente RESTART entsprechen der Anforderung nach einem schnellen, sicheren und weitgehend automatisierten Wiederanlauf. Die Sicherheit der Daten ist derzeit durch die Funktionsfähigkeit des Plattenstapels, auf dem sich die Datenbasis befindet, begrenzt.
Bei der Betrachtung des Gesamtsystems Trägerrechner/JQMS sind die Aspekte Leistungsfähigkeit und Zuverlässigkeit interessant. In unserem Falle kann die Leistungsfähigkeit, bezogen auf den Datenanfall, als ausreichend bewertet werden. So beträgt z.B. die reale Leistung für die Lochkarteneingabe ca. 5 LK/sec. Die maximale stündliche Ankunftsrate wurde mit 10 568 LK gemessen. Das würde einer Auslastung von etwa 60% entsprechen. Im Gesamtdurchschnitt ergibt sich eine LKE-Auslastung von ca. 40%. Ähnliche Verhältnisse sind für die Druckerausgabe abzuschätzen.
Wesentlich kritischer ist die Zuverlässigkeitsanforderung zu betrachten. Hier geht neben der sporadischen Ankunftsrate die Jobdurchlaufstrategie ein. Da wir letzlich eine durchgehende Prioritätsstrategie realisieren wollen, ist die Eingabe der Jobs unmittelbar bei Abgabe erforderlich. Damit verlangen wir die ständige Verfügbarkeit der READER-Funktion während der gesamten Öffnungszeit der Jobannahme. Darüberhinaus verlangen garantierte Antwortzeiten entsprechend der Priorität einer Aufgabe die Verfügbarkeit des Trägerrechners für die Komponenten SCHEDULER und WRITER. Die einzelnen Komponenten arbeiten dabei im Multiprogramming. Da wir in diesem Zeitraum, während der Öffnungszeit der Jobannahme, eine unmittelbare Prozeßsteuerung durchführen wollen, sind die Sicherheitsanforderungen auch entsprechend hoch. In den übrigen Zeiten, insbesondere auch an den Wochenenden, werden die Anforderungen vornehmlich durch den Jobdurchlauf bestimmt und es ergeben sich größere, zusammenhängende Zeiträume, in denen die Nichtverfügbarkeit des Gesamtsystems ohne spürbare Auswirkungen auf den technologischen Ablauf bleibt. Die uns zur Verfügung stehende technische Basis des ES 1020 konnte den skizzierten Anforderungen nicht genügen.
Die zum Bediener- bzw. Nutzerverhalten ermittelten Daten bedürfen - auf Grund der relativ geringen Testmöglichkeiten - zu ihrer Bestätigung weiterer Messungen.

Literaturhinweise

1 Barthold, Joachim
Anke, Hartmut

Methoden des relationalen Datenbankkonzepts für die Jobverwaltung in einem Ein- Ausgabe-Verbund
Vortrag zur INFO 81
Veröffentlicht in ZfR-Informationen INFO 81
Berlin-Adlershof 1981

Verfasser:

Dipl.-Ing. Hartmut Anke
TU Dresden
Rechenzentrum
8027 Dresden
Mommsenstraße 13

# Simulationsmodell der Stapelverarbeitung im Multiprogrammbetrieb

Doz. Dr. sc. techn. Gerhard Bergholz
Gabriele Mrosko

## 1. Optimierung durch Variantenvergleich und Nutzensnachweis mit Hilfe der Simulation

In den kommerziellen Rechenzentren werden Projekte der Massendatenverarbeitung, für die verschiedenen Abarbeitungsvarianten (täglich, wöchentlich, monatlich) existieren können, periodisch abgearbeitet.
Ein Projekt umfaßt ca. 50-200 Jobs, die durch die Erzeugung und den Verbrauch von Dateien voneinander abhängig sind.
Die Jobs eines Projektes bilden ein Jobnetz, dessen Abarbeitung durch das erweiterte Programmsystem IEFBR14 automatisch gesteuert wird.
Durch die geeignete Gestaltung der Jobnetze bzw. durch die verschachtelte Abarbeitung mehrerer Jobnetze ist eine Maximierung der Parallelität mit dem Ziel der Leistungserhöhung möglich.
Dabei ist zunächst die Aufstellung eines maximal parallelen Jobnetzes unter Berücksichtigung der logischen Abhängigkeiten der Jobs erforderlich.
Anschließend erfolgt die Anpassung des maximal parallelen Jobnetzes an die entsprechend der Konfiguration vorhandenen Ressourcen (Partitionanzahl, verfügbare periphere Geräte).
Zur Lösung dieser Aufgaben werden die Methoden der Netzplantechnik mit beschränkten Ressourcen herangezogen.
Es hat sich gezeigt, daß die Parallelität ein Maximum annimmt, wenn die Jobs mit der größten Jobverweilzeit den Vorrang erhalten /2/.
Bei bekannten Jobverweilzeiten kann durch Ermittlung der Dauer des kritischen Weges die Projektlaufzeit vorhergesagt werden.
Dabei bleibt die Laufzeitverlängerung der Jobs, die durch die Erhöhung der Parallelität zustande kommt, unberücksichtigt.
Um exakte Aussagen über die voraussichtliche Projektlaufzeit treffen zu können, sind weiterführende Untersuchungen erforderlich.

Diese können mit Hilfe
- einer Projektumstellung und anschließender Systemmessung
- der Nutzung bedienungs- und wahrscheinlichkeitstheoretischer Modelle oder
- der Simulation

durchgeführt werden.
Die erste Möglichkeit ist sehr zeit- und kostenaufwendig und die Ergebnisse sind schwer reproduzierbar.
Die handhabbaren bedienungstheoretischen Modelle setzen ein konstantes Jobprofil (das in unserem Fall nicht gegeben ist) voraus. Es ist nicht möglich, Prioritäten, die auf die Zuteilung der Zentraleinheit Einfluß haben, zu berücksichtigen.
Als Ausweg bieten sich die Simulationsuntersuchungen an.
Neben der voraussichtlichen Leistungserhöhung durch Maximierung der Parallelität können Möglichkeiten der Leistungssteigerung durch verschiedene Auswahlstrategien der ZVE (Zeitscheibentechnik, absolute Prioritäten) überprüft werden.

## 2. Aufbau des Simulationsmodells

Auf der Basis des PS SIMDIS wurde ein Simulationsprogramm entwickelt, das die Stapelverarbeitung unter Steuerung des Betriebssystems OS/ES MFT nachbildet.

### 2.1. Eingangsgrößen für das Simulationsprogramm

Dem Programm sind folgende Eingangsgrößen zur Verfügung zu stellen:

| | |
|---|---|
| - Partitionaufteilung | (Partitionnr., zugeordnete Jobklassen) |
| - Konfiguration | (Zahl der verfügbaren Geräte je Geräteklasse) |
| - Struktur des Jobstapels | (Abhängigkeitsmatrix) |

Für den nachzubildenden Job werden folgende Daten gefordert:

| | |
|---|---|
| - Jobklasse | (entsprechend CLASS-Parameter) |
| - Startpriorität | (entsprechend PRTY-Parameter) |
| - Bedarf an fest zugeordneten Ressourcen | (Zahl der Ressourcen einer Geräteklasse, die dem Job während der gesamten Laufzeit zugewiesen sind) |
| - mittlere Bedienungszeit je Zugriff | (Zeit für die Verarbeitung eines physischen Satzes) |

- Zugriffshäufigkeiten zu den peripheren Geräten.

2.2. Beschreibung des Simulationsmodells

Zur Darstellung des simulierten Modells bietet sich die Forderungslaufplantechnik nach Bergholz /1/ an. Der Forderungslaufplan kann mit geringem Aufwand in einen SIMDIS-Aktivatorlaufplan umgesetzt werden.
In der Abbildung 1 ist der Forderungslaufplan (FLP) für die Abarbeitung eines einfachen Jobnetzes gegeben.
Die grafische Darstellung ist folgendermaßen zu interpretieren:
Die Forderungsquelle erzeugt für jeden nachzubildenden Job einen Aktivator.
Alle abhängigen Jobs verbleiben bis zur Freigabe nach der ordnungsgemäßen Beendigung ihrer Vorgänger in der HOLD-Kette (Job 2 - Job 4).
Die unabhängigen Jobs (Job 1) reihen sich in die ihrem CLASS-Parameter entsprechende Jobwarteschlange ein und warten auf die Zuweisung eines geeigneten Hauptspeicherbereiches.
Für die Jobs einer Klasse erfolgt die Vergabe der Partitions entsprechend dem PRTY-Parameter.
Bei Freiwerden eines Hauptspeicherbereiches wird für den ausgewählten Job überprüft, ob alle benötigten Ressourcen verfügbar sind. Ist das nicht der Fall, wird dieser Job suspendiert und ein anderer zur Verarbeitung freigegeben.
In der Abbildung 2 ist der FLP für die Abarbeitung eines Jobs im Multiprogrammbetrieb gegeben.
Nach der Zuweisung aller fest zugeordneten Ressourcen stellt der Job abwechselnd Bedienungsanforderungen an die Informationsaustauschressourcen (IAR) und die ZVE.
Bei den IAR sind die geteilt genutzten IAR und die dem Job fest zugeordneten Ressourcen zu unterscheiden.
Zu den geteilt genutzten IAR zählen neben den Systemdateien alle Dateien des Modus SHR, die Selektorkanäle und die zugehörigen programmtechnischen Ressourcen.
Zu den fest zugeordneten Ressourcen gehören alle peripheren Geräte, die über den Multiplexkanal angesprochen werden. Die Wartezeiten, die vor diesen Ressourcen entstehen, können vernachlässigt werden.

Die Wahrscheinlichkeit für den Zugriff zu einer IAR kann man durch die relative Zugriffshäufigkeit annähern.
In den IAR werden die Jobs entsprechend der Reihenfolge ihrer Ankunft (FIFO) bedient.
Nach jedem Informationsaustausch folgt ein Verarbeitungsschritt.
In dem Simulationsmodell werden die beiden Möglichkeiten, die im OS/ES für die Vergabe der ZVE existieren, berücksichtigt. Durch Eingabe eines Parameters kann man steuern, welche der Varianten abzuarbeiten ist.
Die absoluten Prioritäten für die Vergabe der ZVE sind analog dem OS/ES den Programmbereichen fest zugeordnet.
Bei der Nutzung der Zeitscheibentechnik ist die Größe der Zeitscheibe variabel und dem Programm zur Verfügung zu stellen.
Alle Bedienungszeiten einer Forderung werden als exponentiell verteilt angenommen.
Der Wechsel von Informationsaustausch und Verarbeitung wird pro Job sooft durchlaufen, wie physische Datensätze zu verarbeiten sind.
Aus der Abbildung 2 ist ersichtlich, daß die Abarbeitung eines Jobs sequentiell erfolgt.
Nach Beendigung eines Jobs werden alle ihm fest zugeordneten Ressourcen freigegeben. Die Nachfolgerjobs können die HOLD-Kette verlassen und sich in die Jobwarteschlange einreihen.
Alle Bedienungszeiten der Aktivatoren und die Verweilzeit werden erfaßt. Die Meßstellen sind aus den Abbildungen 1 und 2 ersichtlich.

2.3. Ergebnisse der Simulation

Als Ergebnisse der Simulation wurden für jeden Jobs folgende Größen ausgegeben.

- Jobnr.
- Partitionnr.
- Startzeit
- Stopzeit
- Bedienungszeit
- Verweilzeit
- Verweilzeitfaktor (Maß für Laufzeitverlängerung eines Jobs im Multiprogrammbetrieb gegenüber dem Solobetrieb)
- durchschnittliche Anzahl der Unterbrechungen für eine ZVE-Bedienung.

Der letztgenannte Wert ist von Interesse, um einen Vergleich zwischen der Vergabe der ZVE nach absoluten Prioritäten bzw. nach Zeitscheibentechnik durchzuführen und die optimale Auswahlstrategie für eine Projektabarbeitung festzulegen.
Für jeden nachzubildenden Jobstapel werden neben der Start- und Stopzeit der Verweilzeitfaktor, der Leistungsfaktor (als Maß für die bewältigte Bedienungszeit im Beobachtungszeitraum), die Auslastung aller Ressourcen und die Beschäftigungskoeffizienten aller Ressourcen ausgedruckt.

Nachweis der Adäquatheit des Simulationsmodells

Es ist zu überprüfen, inwieweit bei gleichen unabhängigen Einflußfaktoren die durch die Messung realen System gewonnenen Bewertungsgrößen mit den durch die Anwendung des Modells gewonnenen Bewertungsgrößen übereinstimmen.
Die Grundlage für den Adäquatheitsnachweis bildet ein Benchmark-Test, der durch die GTF-Routinen erfaßt wurde.
Zu dem Vergleich wurden folgende Bewertungsgrößen herangezogen:

- die Auslastungsgrade aller Ressourcen
- der Leistungsfaktor während der Abarbeitung des Stapels.

Ein Systemjob bildet die durch den Overhead erzeugte Arbeitslast nach. Für diesen Job wird während der Simulation eine Inaktivitätsphase eingeführt, um zu garantieren, daß während des gesamten Beobachtungszeitraumes das System aktiviert werden kann.

In der Tabelle 1 ist eine Gegenüberstellung der Ergebnisse des Simulationsversuches mit denen des Benchmark-Tests enthalten. Die maximale Abweichung der Bewertungsgrößen beträgt rund 7 %. Dieses Resultat berechtigt zu der Annahme, daß das Simulationsmodell zur Nachbildung der Stapelverarbeitung im Multiprogrammbetrieb genutzt werden kann.

## 4. Beispiele für die Anwendung des Simulationsmodells

Mit Hilfe der Simulation wurden mehrere Projekte des DVZ Dresden, während deren Abarbeitung die Auslastung der Ressourcen relativ gering ist, auf die Möglichkeit einer Leistungssteigerung durch Erhöhung der Parallelität innerhalb des Projektes oder durch parallele Verarbeitung mit einem anderen Projekt, überprüft. Der Vergleich der mittels Simulation nachvollzogenen realen Abarbeitung mit der optimierten Abarbeitung ergab für die Erhöhung der Parallelität innerhalb eines Projektes die in der Tabelle 2a zusammengestellten Bewertungsgrößen.
Die Tabelle 2b zeigt die Bewertungsgrößen für die parallelisierte Abarbeitung zweier Projekte, die bisher nacheinander abgearbeitet wurden.
Ein Vergleich der Leistungsfaktoren zeigt, daß die größten Reserven für eine Leistungserhöhung bei der Parallelisierung der Abarbeitung mehrerer Projekte liegen.

## Literaturverzeichnis

/1/ Bergholz, G.: Verhaltensmodelle von Prozeßrechnern
Akademie-Verlag Berlin 1980

/2/ Bergholz, G.: Leistungsanalyse und Leistungserhöhung von Rechenanlagen, edv aspekte, Heft 2/1982, S. 2-6

Tabelle 1: Gegenüberstellung der Bewertungsgrößen während des Benchmark-Tests mit den Simulationsergebnissen

| | Simulation | Benchmark-Test | Differenz |
|---|---|---|---|
| $\eta_0$ | 0,015 | 0,014 | 7,1 % |
| $\eta_1$ | 0,369 | 0,345 | 6,9 % |
| $\eta_2$ | 0,121 | 0,127 | 4,7 % |
| $\eta_{ZVE}$ | 0,902 | 0,955 | 5,5 % |
| $R$ | 1,41 | 1,41 | - |

$\eta_0$ - Auslastungsgrad IAR 0
$\eta_1$ - Auslastungsgrad IAR 1
$\eta_2$ - Auslastungsgrad IAR 2
$\eta_{ZVE}$ - Auslastungsgrad ZVE
$R$ - Leistungsfaktor

Tabelle 2a: Durch Erhöhung der Parallelität innerhalb eines Projektes erreichbare Verbesserungen

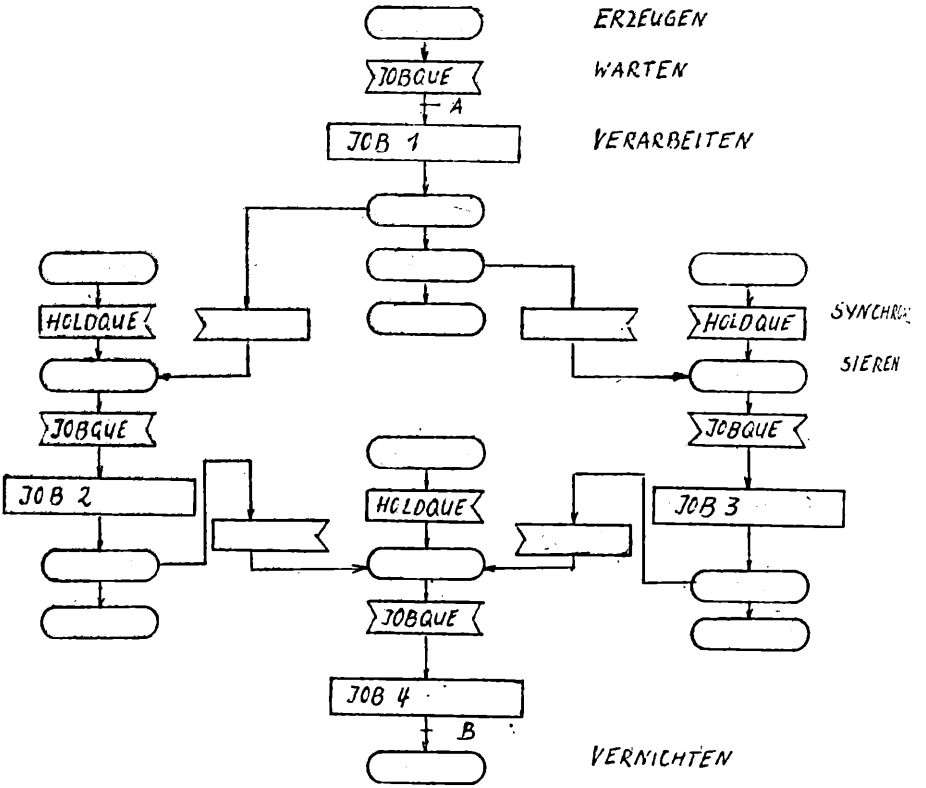
| | | $T_h$ | M | F | R |
|---|---|---|---|---|---|
| Projekt 1 | reale Abarbeitung | 2138s | 2,0 | 1,61 | 1,25 |
| | optim. Abarbeitung | 2071s | 2,5 | 1,82 | 1,41 |
| Projekt 2 | reale Abarbeitung | 10 402s | 1,94 | 1,39 | 1,39 |
| | optim. Abarbeitung | 8 007s | 2,5 | 1,48 | 1,68 |

Tabelle 2b: Durch die parallele Verarbeitung mehrerer Projekte erreichbare Verbesserungen

| | | $T_h$ | M | F | R |
|---|---|---|---|---|---|
| Projekt 3 | reale Abarbeitung | 2668s | 1,83 | 1,22 | 1,5 |
| Projekt 4 | optim. Abarbeitung | 2094s | 2,77 | 1,46 | 1,9 |

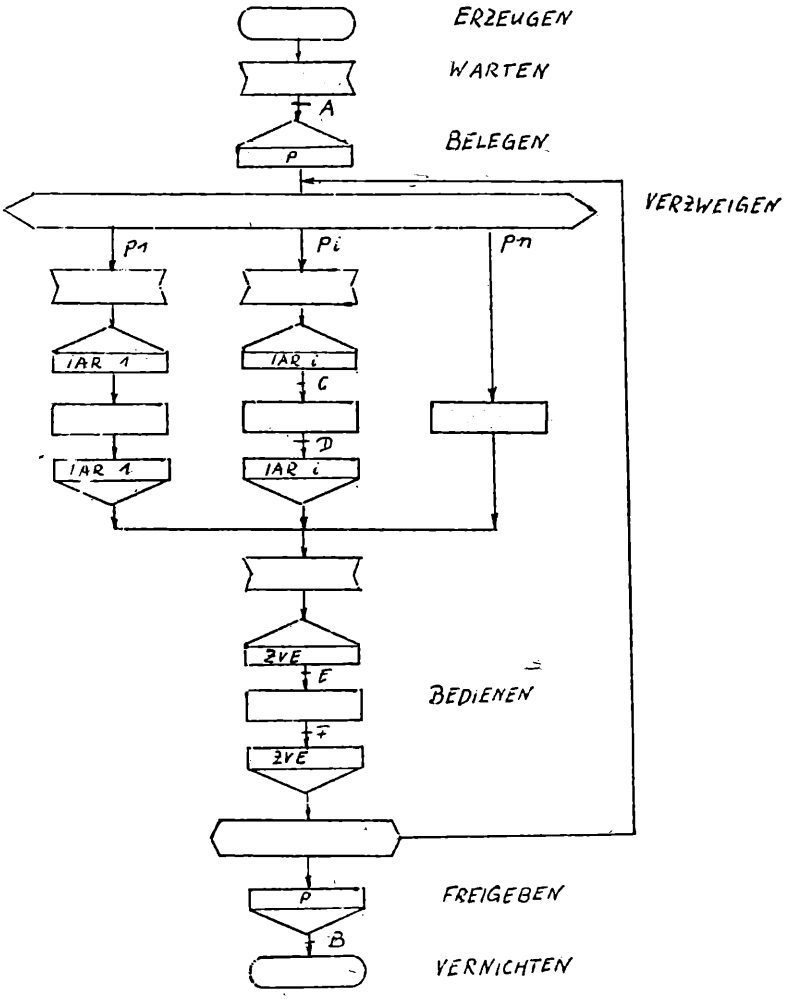
$T_h$ - Beschäftigungszeit für die Abarbeitung eines Jobstapels
M - Multiprogrammfaktor
F - Verweilzeitfaktor
R - Leistungsfaktor

Abbildung 1: FLP für die Abarbeitung eines einfachen Jobnetzes




$\overline{AB}$ - Beschäftigungsziel des Jobstapels

Abbildung 2: FLP für die Abarbeitung eines Jobs im Multiprogrammbetrieb




$\overline{AB}$ - Verweilzeit eines Jobs

$p_i$ - Verzweigungswkt. zu der IAR i

$\overline{CD}$ Bedienungszeit einer Forderung in IAR i

$\overline{EF}$ Bedienungszeit einer Forderung in der ZVE

Autorenverzeichnis


Aden, H. 342
Anke, H. 516
Barrera, J. 384
Bauer, G. 284
Bazewicz, M. 124,428
Bergholz, G. 105,524
Bertholdt, D. 281
Böhme, P. 356
Bothe, K. 220
Debbwy, J. 428
Detering, R. 97,179
Fellien, A. 415
Fenske, B. 362
Fichtner, N. 446
Haagen, E. 352
Haberland, H. 330
Hagen, M. 370,436
Herrlich, O. 97
Heymer, V. 136
Hohberg, B. 221
Horn, C. 245
Jantke, R. 462
Kelemen, J. 313
Kieschke, R. 291
Klaembt, E. 370,436
Köhli, S. 7
Kollár, L. 308
Kosciolowicz, R. 231
Kroß, G. 485
Meißler, H.-G. 362,491
Melzer, J. 454
Meuche, H.-F. 298
Mika, T. 124
Mikulecky, P. 313
Mrosko, G. 524
Nikulski, E. 276
Pester, M. 408
Pfeiffer, P. 187
Pi[illegible]s, C. 498
Pilgrim, J. 71
Polze, C. 215
Richter, E. 199
Roscher, G. 330
Sacklowski, U. 169
Sadowski, H. 478
Schirmbacher, P. 370,436
Schirrow, R. 326
Schreiber, E. 468
Schreiter, G. 155
Stantschev, V. 179
Tauscher, W. 199
Trautloft, R. 209
Vack, G.-U. 268
Vierke, B. 352
Wagner, K. 420
Wegener, E. 298
Weikmann, C. 503
Wenzlaff, B. 58
Winks, H. 40
Wittig, W. 379
Wörfel, F. 508